An Estimation Theory Approach to
Detection and Ranging of Obscured Targets in 3-D LADAR Data

THESIS

Charles R. Burris, First Lieutenant, USAF

AFIT/GE/ENG/06-10

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

AFIT/GE/ENG/06-10

An Estimation Theory Approach to Detection and Ranging of Obscured Targets in 3-D LADAR Data

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Charles R. Burris, B.S.E.E.

First Lieutenant, USAF

March 2006

AFIT/GE/ENG/06-10

An Estimation Theory Approach to
Detection and Ranging of Obscured Targets in 3-D LADAR Data

Charles R. Burris, BS

First Lieutenant, USAF

Approved:

_____/Signed/_____ _____
Stephen C. Cain (Chairman) Date

_____/Signed_____ _____
Matthew E. Goda, Lt Col, USAF (Member) Date

_____/Signed/_____ _____
Richard D. Richmond (Member) Date

Acknowledgments

I would like to express my sincere appreciation to my faculty advisor, Dr. Stephen Cain, for his guidance and support throughout the course of this thesis effort. The insight and experience was certainly appreciated. I would, also, like to thank my sponsor, Rich Richmond, from the Air Force Research Lab for both the support and latitude provided to me in this endeavor. I would also like to thank my wife for her support and understanding of my peculiar behavior during this process. Most of all I would like to thank God for helping me to pull through.

Charles R. Burris

Table of Contents

Page

List of Figures

Page

List of Tables

Page

AFIT/GE/ENG/06-10

Abstract

The purpose of this research is to develop an algorithm to detect obscured images in 3-D LADAR data. The real data used for this research was gathered using a FLASH LADAR system under development at AFRL/SNJM. The system transmits light with a wavelength of 1.55 micrometers and produces 20 128 X 128 temporally resolved images from the return pulse separated by less than 2 nanoseconds in time. New algorithms for estimating the range to a target in 3-D FLASH LADAR data were developed. Results from processing real data are presented and compared to the traditional correlation receiver for extracting ranges to the target. This research shows that the algorithms presented are capable of distinguishing two surfaces separated by only 40 inches using real data.

An Estimation Theory Approach to
Detection and Ranging of Obscured Targets in 3-D LADAR Data

I. Introduction

One of the most intriguing problems in laser imaging is the detection and ranging of obscured targets in battlefield operations. 3-D Light Detection and Ranging (LIDAR) or Laser Detection and Ranging (LADAR) sensors are both capable of performing this task. For clarification, LADAR sensors are LIDAR sensors, only the term LADAR refers specifically to laser light. Most current methods of detecting obscured targets with 3-D LIDAR sensors require scanning the scene, registering the images, and compiling them as accurately as possible (Murray,2003). The goal of this thesis is to produce an algorithm that will detect obscured targets and provide ranging information from a single laser pulse. The ability to gain this information with a single pulse decreases the required time of both getting and processing the images and complexity of the image processing required.

Enemy targets are often obscured by camouflage netting to prevent them from being detected and destroyed. Lives can be lost when enemy targets are not detected promptly and accurately. One method of obscured target ranging is from the trajectory of artillery from the target. The trajectory method is not very accurate and it requires having already been shot at. For these reasons, the detection and identification of obscured targets has been a goal of military sensing for a long time.

1.1 Background

Applications of Light Detection and Ranging (LIDAR) imaging devices for military purposes include the detection and ranging of obscured targets such as tanks hidden by camouflage netting or heavy tree canopies, target identification, detection of mines both on land and submerged in water, and mapping. There are other uses as well, robotic vision for example. The ability of 3-D LIDAR to retrieve range information along with the images is advantageous for military operations, as it is the key to improving the detection of obscured targets and the proper identification of targets.

3-D LIDAR sensors send out pulses of light and produce a series of images that create a 3-D scene from the variations in the intensity of light returned from the pulse. Single-pulse time-of-flight Flash LIDAR sensors create this 3-D scene by sampling in time the number of photons returned to each pixel (Halmos, 2003; Murray, 2003). The variation in the number of photons received or light intensity at each pixel creates the 2-D image. The sampling in time of the returned pulse of light creates a series of 2-D frames where each frame represents the intensity of light received at that moment in time. The collection of these 2-D frames produces the 3-D image. 2-D images produced from light returned by closer objects will appear in earlier frames while images from light returned by objects that are farther away will appear in later frames as shown in Figure 1.1.

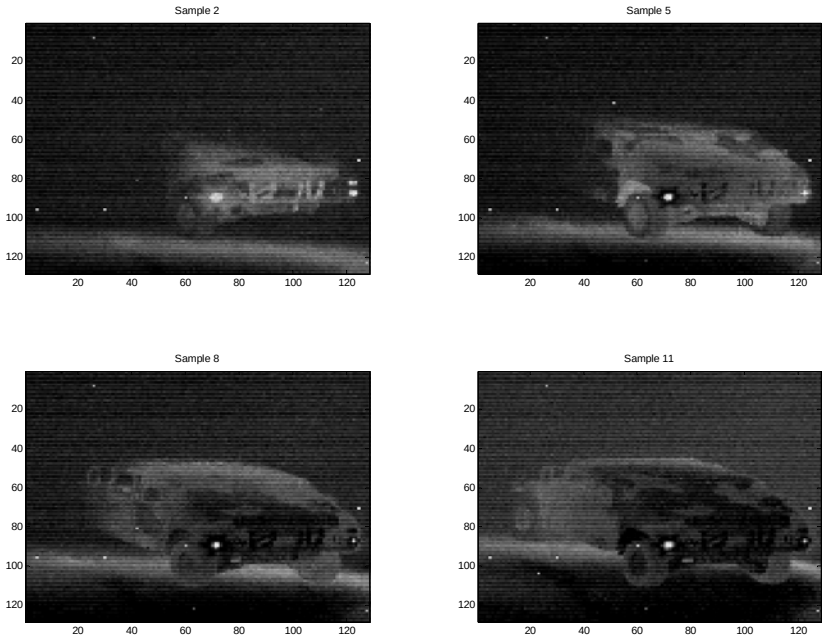

Figure 1.1 illustrates the 3-D nature of the images obtained by a 3-D FLASH LIDAR system with images of a humvee taken with the sensor under development by AFRL/SNJM. The images in this figure are 3 samples apart in time.

Multiple pulse LIDAR sensors combine information from multiple pulses to recreate the scene as accurately as possible. The use of multiple pulses allows for an averaging of the noise, which reduces the overall noise intensity. However, the averaging may cause distortion or blurring of the features in the image (Armstrong, 2004; MacDonald, 2004). If the scene changes or the sensor moves even slightly, then the images must be realigned to prevent distortion. The movement of the scene, sensor, or target creates a new source of noise and increases the time and difficulty of the image processing required for obtaining useful data.

Scanning LADAR uses multiple pulses to create a 3-D scene while slowly changing the look angle. The scanning of the scene creates multiple images with full

range information but only partial pieces of the spatial information. These images are then pieced together to create one spatially larger 3-D scene. One method of scanning LADAR, used in the Jigsaw program, involves flying over the scene multiple times with a 3-D LADAR sensor. The idea is that each pass will provide a different view of the scene so that pieces of the target that were hidden from view on an early pass may be seen on one of the later passes as a result of the new observation point. The task of properly registering the images is complicated by the conditions under which this information is gathered; the movement of the aircraft, the need to know exactly where the aircraft is and how it is oriented, and the inability to fly the desired route for the desired observation. This method could also be very dangerous in the case of a hostile environment.

Fixed multiple-pulse LADAR sensors may change the time at which the return pulse is sampled to vary the range of the image produced. This technique is useful for the imaging of multiple objects separated primarily by range. It allows for imaging of two objects separated in range by a larger distance than the sensor can accurately image with a single pulse.

Currently most types of LIDAR systems use the same LIDAR processing algorithm for extracting the range from sensor to target. This algorithm is the correlation receiver (Cain, 2004; Gelbart, 2003; Halmos, 2003; Murray, 2003; Walter, 2005). This research proposes a new algorithm that is discussed in Chapter 3.

1.2 Resolution Limits

Range resolution and spatial resolution are the two types of resolution that most important to 3-D LADAR imaging devices. According to Khoury (2005), there is a constant volume of resolution (VOR), and improving range resolution or spatial resolution comes at the expense of decreasing the other. LADAR images can be presented as a data cube or 3-D matrix. The spatial information is stored along the X and Y-axes having line number "m" and "n" respectively where each line is spatial resolution. The range information is stored along the Z-axis having "l" lines of resolution. Figure 1.2 below provides a graphical illustration. Khoury (2005) defines the VOR as the multiplication of the number of lines in the X, Y, and Z directions. Based on the concept of a constant volume of resolution it follows that increasing the number of lines in one direction, increasing the resolution in that direction, would come at the expense of decreasing the number of lines in one of the other directions. The units of resolution for a volume of resolution are referred to as voxels where a voxel is one unit in each of the X, Y, and Z directions.

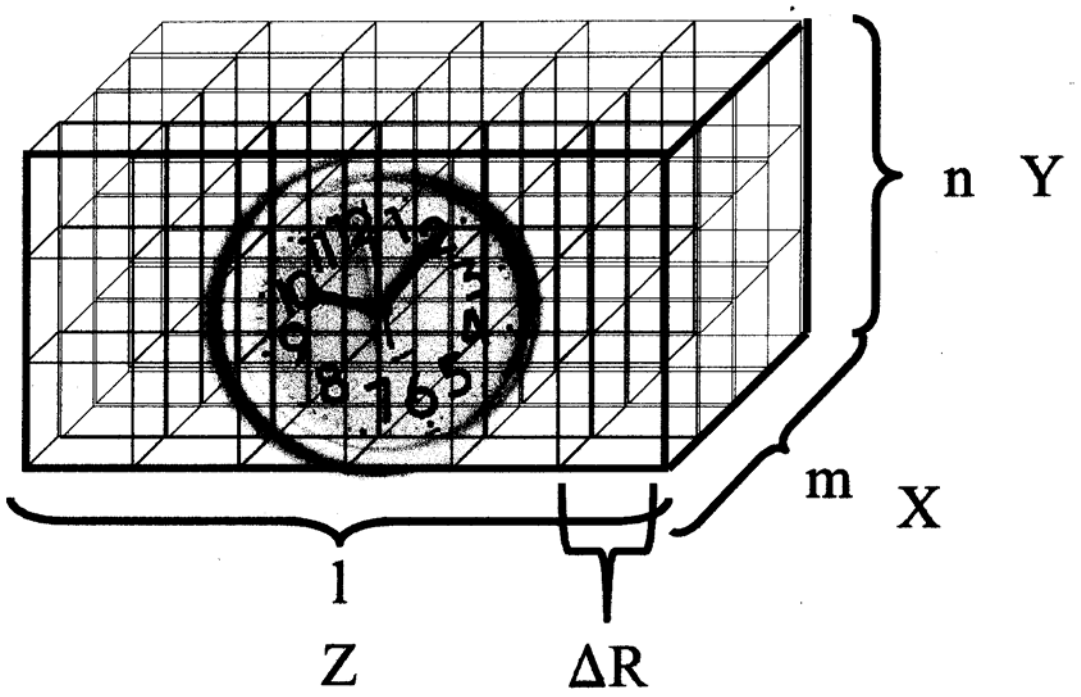

Figure 1.2: 3-D LIDAR resolution cube (Khoury, 2003).

1.3 Detection of Obscured Targets

The detection and identification of obscured targets has been a goal of military sensing for a long time. "Recently, the Defense Advanced Research Projects Agency (DARPA) and the U. S. Army's Future Combat Systems (FCS) began a serious effort to develop and field advanced 3D ladar technology and hardware to address the need to detect and identify targets under heavy foliage and camouflage" (Murray, 2003). The fundamental concept of this program, Jigsaw, is to register the target by adding small segments that are not obscured by taking advantage of the small voids of camouflage or foliage. This process is repeated as the sensor is moved to unveil additional segments of the target. Eventually, enough information is obtained to produce an image of the target

when the individual frames are fused into one composite image. This process is illustrated below in Figure 1.3.

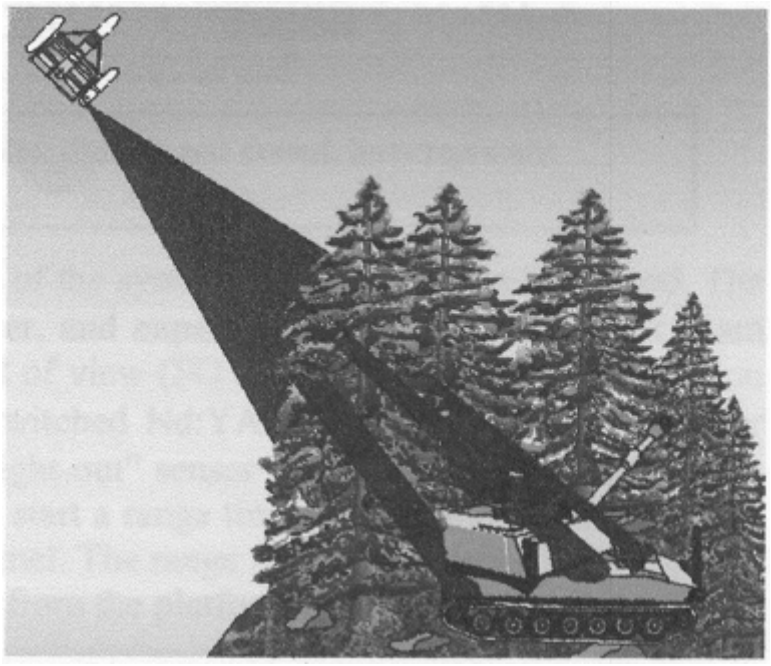

Figure 1.3: Multi-aspect 3-D flash LADAR from a small UAV platform (Murray, 2003).

1.4 Scope

The goal of this research is only to design an algorithm to detect and range obscured targets from a single pulse of laser light. This algorithm may be adaptable to be incorporated multiple pulse methods or even combined with other single pulse algorithms, but that will not be addressed in this research. This research will not seek to redesign or optimize the sensor in terms of physical or optical design. Although this research is being conducted with the goal of designing an algorithm to be used with a specific sensor under development at AFRL/SNJ, the algorithm will be designed generically so that it may be tailored to be used with other LIDAR sensors.

1.5 Approach/Methodology

The new algorithm will first be tested on real data with a known un-obscured target or one-surface model. The target is a 3-D box with holes cut out of the front surface. Pixels can be chosen in the data to represent a surface at the front of the box or a surface at the back of the box. After acceptable performance is achieved on the data with an un-obscured target, the algorithm will be tested on real data with an obscured surface or two-surface model. The target will be modified by replacing the front surface of the target with a semi-transparent material to create the effect of a hidden target. This is referred to as the two-surface model because light will be reflected from the semi-transparent surface and the back surface of the target.

The time sampling of light from one pixel of the one-surface model should show one local maximum. The two-surface model should have two local maxima. The first local maximum in time will be the semi-transparent surface while the second local maximum will be the actual target. The new algorithm uses a gated parabola with the same width as the pulse width of the light transmitted by the laser. For the one-surface case the amplitude and center of the parabola have to be solved for to best estimate the actual data. The noise bias must also be estimated. The algorithm does this by setting the derivative with respect to the amplitude of the sum-squared error between the real data and the generic parabola used to model the returned pulse to zero and solving for the amplitude that generates the smallest error for the hypothesized location of the surface. This is repeated for each possible hypothesized location where the parabola could be centered. Meanwhile the bias term is being calculated by taking the average of the values

everywhere that the parabola is not located. Then the location for the center of the parabola which produced the least error is chosen as the range estimate. The algorithm for the two-surface case has two additional parameters to solve. They are the amplitude and location of the second surface, which in the case of a camouflaged tank would be the tank.

II. Sensor Model

This chapter describes the setup of experiments conducted by AFRL/SNJM, which were designed to test the imaging and ranging capabilities of the 3-D Flash LADAR system under development. Section 1 will explain the experimental setup. Section 2 will go through calculations for the expected signal in a typical pixel. Figure 2.1 shows a conceptual diagram of the experimental system.

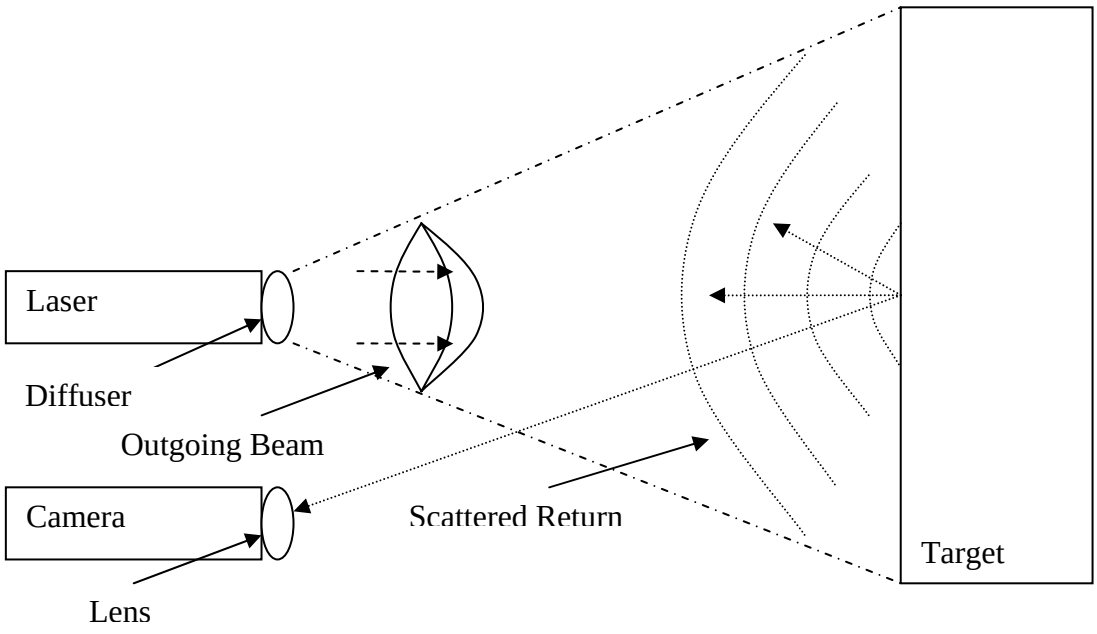

Figure 2.1 shows a conceptual diagram of the system.

2.1 Experimental Setup

This section will describe the experimental setup starting with the laser and diffuser. The 3-D Flash LADAR system used in this study uses a 70mW Big Sky laser with a diffuser. The laser transmits a pulse of light with a wavelength of 1.55μm corresponding to a frequency of 193 THz. The benefits of transmitting light at this

frequency and low power are that it is beyond the visible range, so the enemy cannot see it, and it is eye-safe. Since the outgoing pulse exits the laser cavity in a tightly focused beam, it must be passed through a diffuser to ensure that the pulse spreads enough to completely illuminate the target.

2.2 Target Description

The target used in this experiment is a plywood box. The front surface has squares and rectangles cut out of it so that the back surface can be seen through the cut-outs. The back surface is solid. The left side is 12 inches deep and the right side is 40 inches deep. The target is painted white to increase light reflectivity. The figure below is an actual photograph of the target.

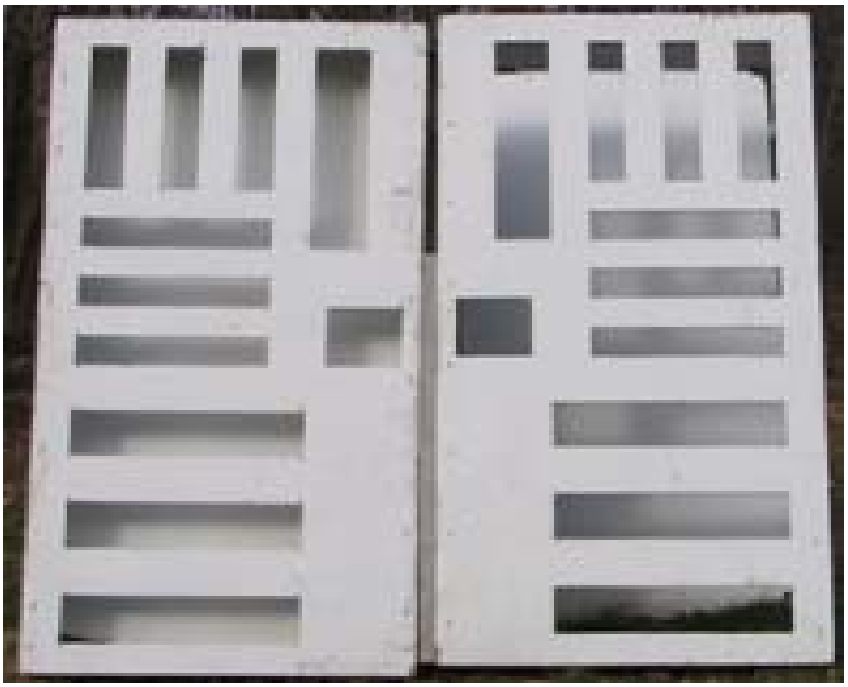

Figure 2.2 shows an actual photograph of the target.

2.3 Camera

The camera consists of receiver optics, light filter, detector array, and a Read Out and Integration Circuit (ROIC). The receiver optics are designed to transform the spherical wave returned from the pulse into points. The light filter is used to reduce the amount of non-laser produced light allowed to enter the camera. Any light from the environment or any source other than the laser will contribute to noise. The detector array consists of a 128 X 128 micro-lens array and a 128 X 128 pixel array. The micro-lens array is used because the pixel array has a low fill factor. The micro-lens array creates the effect of a higher fill factor. This is accomplished by focusing light from an area corresponds to the pixel size that would be needed to create nearly a 100% fill factor by focusing that light on to the smaller pixel size used in the detector array. This is illustrated in Figure 2.3. The ROIC consists of a bank of capacitors, amplifiers, and A/D converters. Each pixel is connected to a capacitor for each sample of the pulse collected (Cain, 2004), one amplifier, and one A/D converter as shown in figure 2.4.

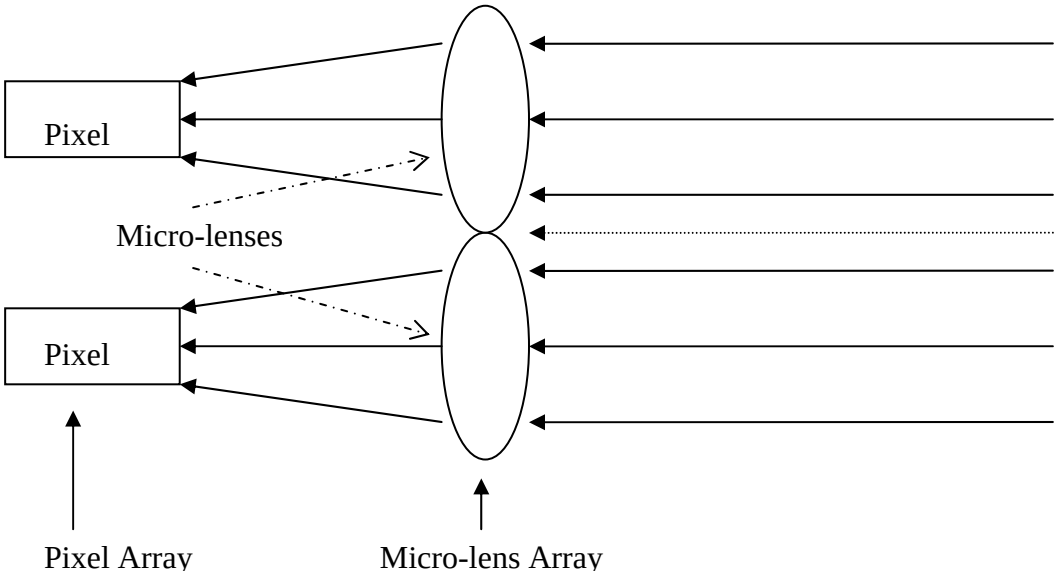

Figure 2.3 shows how the micro-lens array refocuses the light to increase the number of photons received by the pixel.

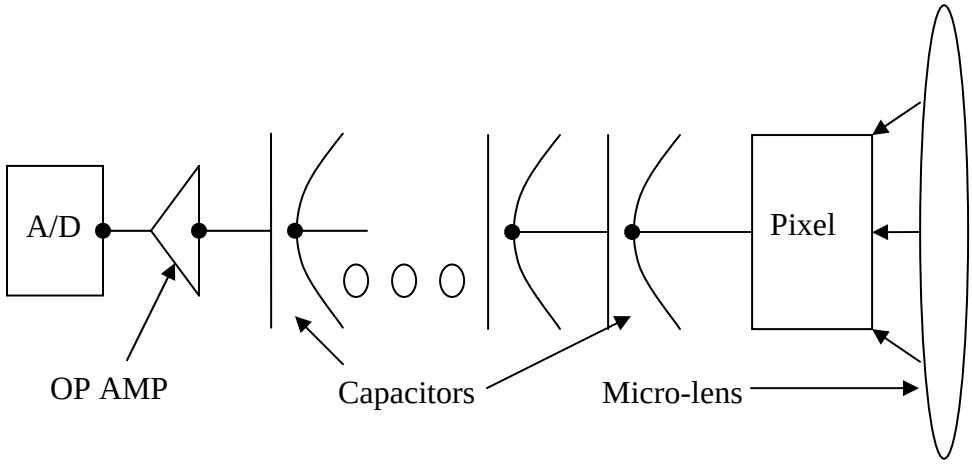

Figure 2.4 shows the circuitry connected to the pixel.

2.4 Photons to Digital Count Calculations

This section will start at the laser and calculate the number of photons expected at each stage of the process to produce an estimate of the signal at the detector. The first section will calculated the total number of photons per outgoing pulse. The second section will calculate the number of the outgoing photons per pulse that will illuminate the area that corresponds to one pixel at the target. The third section calculates the number of photons per pulse received by the camera for a single pixel. The last section will calculate the expected digital count output by the sensor based on the previous calculations.

2.4.1 Total Number of Photons per Outgoing Pulse

The power output (P_{out}) of the laser is 70 millijoules. Its pulse repetition frequency (prf) is 10 pulses per second. The number of photons per pulse (P_p) is:

(2.1)
$$P_p = \frac{joules / pulse}{E_P}$$

$E_P = h*C/\lambda$ where E_P is the energy contained in a photon, h is Planck's constant, C is the speed of light, and λ is the wavelength. The complete equation for the total number of photons per outgoing pulse is:

(2.2)
$$\frac{photons}{pulse} = \frac{\frac{P_{out}}{prf}}{h*\frac{C}{\lambda}}$$

The numeric result of this calculation is $5.46*10^{17}$ photons per pulse.

2.4.2 Number of Photons per Pulse per Pixel

Given that $5.46*10^{17}$ photons per pulse are sent out, we will assume 46% of the pulse will be in view of the camera. This assumption is based on one standard deviation from maximum value of a Gaussian pulse containing 34% of the energy and that the camera is set to receive one standard deviation in each direction. This figure is doubled to account for being on either side of the center of the pulse. This result is squared to account for the Gaussian being 2-D. Applying this 46% (%view) to our previous result yields $2.51*10^{17}$ photons in the view of the camera. The camera has 128 X 128 pixels. Dividing our previous result by the number of pixels results in $1.53*10^{13}$ photons per pixel at the target. Losses of the pulse due to transmission through the air are accounted for in the next step. In this experiment the pulse is traveling through approximately 80 meters of atmosphere the percent transmission (τ_a) of light having a wavelength of 1.55 μm is approximately 100% as the path is short and 1.55 μm is in an atmospheric window. The complete equation for the number of photons hitting the area of a pixel at the target is:

$$\frac{photons}{pixel_T} = \frac{\tau_a * \%view * \frac{photons}{pulse}}{n * m} \tag{2.3}$$

At this point there are $1.53*10^{13}$ photons per pixel at the target.

2.4.3 Number of Photons per Pulse per Pixel Received by the Camera

When the pulse hits the target, only a percentage of it is reflected. This is determined by the reflectivity of the target (r_T). Since the target is plywood that has been painted white a value of 10% is approximated for the target reflectivity. The percentage that is reflected by the target is scattered after hitting the target. This scattering is distributed on a sphere having an area of $2\pi Z^2$ where Z is the range in meters. The target is approximately 80 meters from the sensor. The transmission of the atmosphere is applied again as the photons must return through the atmosphere as they travel from the target to the camera. The number of photons per m^2 is then multiplied by the area of the lens of the camera, $area_L$, to produce the number of photons received by the camera.

(2.4) $$area_L = \pi * \left(\frac{.03}{2} \right)^2$$

(2.5) $$\frac{photons}{pulse_C} = \frac{area_L * \tau_a{}^2 * \%view * \frac{photons}{pulse}}{2\pi * Z^2 * n * m}$$

This calculation results in 269,470 photons received by each pixel per pulse (Khoury, 2005).

2.4.4 Digital Count

The camera is an electron-counting camera. The manufacturer of the camera estimates the conversion percentage from photons to electrons for the camera (CR_c) to be 30%. The capacitors take samples of the number of electrons. The electrons are stored in

the capacitors until the op amp amplifies them and then the A/D converter converts them to digital counts.

(2.6) $$digital\_count = CR_C * \frac{photons}{pulse_C}$$

The digital count expected to be received by one pixel from a single pulse is 80,840. In real data containing a pulse returned from a single surface, the total digital count is around 9800. There are several factors that could attribute to differences between the calculated digital count and the observed digital count. The atmospheric transmission is less than 100%, but we do not know exactly what it was when the data was collected. The reflectance of the target may be more or less than 10% and any losses due to the micro-lens array were unaccounted for. Some photons will be absorbed by the lens and some of the photons will still miss the pixel even after passing through the lens. These factors were not accounted for and could easily account for the differences.

III. One-Surface Algorithm

In this chapter, a new algorithm for processing 3-D Flash LADAR data containing one surface will be discussed and compared to the existing algorithms. The first section will go through the algorithm derivation. Sections 2 and 3 will discuss results from simulated and real data respectively.

3.1 Algorithm Derivation

There are two types of algorithms that are widely accepted for processing Flash LIDAR imagery, peak detection and correlation. Detection and ranging of un-obscured targets can easily be achieved with the one-surface detection algorithm as the light is hitting a single surface and being reflected back to the sensor. One pulse is transmitted and a fraction of that pulse is received after reflecting off the target. If there are two surfaces, the return would be two pulses. The first would correspond to a closer surface and the second to a surface that is farther away. The sum of the number of photons contained in these two return pulses would be less than or equal to the number of photons contained in a single pulse returned from a single surface with the same properties. The two-surface case will be further discussed in Chapter 4.

The new one surface algorithm uses a gated negative parabolic curve as a model to approximate the shape of a pulse returned from a single surface and fits it to the data on a least squares basis. The curve is gated to prevent negative values and reduce edge effects. When the pulse is too close to either end of the data, the rectangle function, which is used to gate the parabola, is shortened to prevent it from adding error. This allows the algorithm to accurately estimate the center location of pulses much closer to the edges of the data than the correlation method or the peak detection method as shown

in Figure 3.3. The one surface algorithm uses a least squares solution approach to a parabola based model. In this approach, a gated parabola, p(x), is used to approximate the shape of the pulse of laser light transmitted by the laser. It then scales and slides the gated parabola until it minimizes the sum-squared difference between the real data and the algorithm's range estimate.

(3.1) $$p(x) = A\left(1 - \frac{(x-m)^2}{w^2}\right) \bullet rect\left(\frac{(x-m)}{2w}\right) + B$$

where A represents the amplitude or gain, w corresponds to the width of the pulse, and m is the set of numbers that represent the ranges that will be tested. B is the signal bias.

For each center (m), the amplitude of the parabola (A), and the signal bias of the data (B), have to be solved for to best estimate the actual data, d(x). First, the signal bias is found from the set of data outside of the rectangle function determined by the current value for m under consideration.

(3.2) $$B = \frac{\sum_{x_1} d(x)}{M_1}$$

The set x_1 is the set of integers over which the rectangle function is equal to zero. The variable M_1 is the number of integers in the set x_1. After the signal bias is calculated for a given m, the best amplitude (A) is calculated by minimizing the sum squared error for a pulse centered at that m. The square error between the data and the estimate is:

(3.3) $$E(m, w, A, B) = \sum_x \left[\left(d(x) - p(x \mid m, w, A, B) \right)^2 \right]$$

Here d(x) is the actual data.

Expanding the right side results in:

$$E(m,w,A,B)=\sum_x\left[\left(d(x)-2d(x)\bullet p(x\mid m,w,A,B)+p(x\mid m,w,A,B)^2\right)\right] \tag{3.4}$$

Substituting in for $p(x\mid m,w,A,B)$ yields:

$$E(m,w,A,B)=\sum_x\left[\begin{array}{l} d(x)-2d(x)\bullet\left(A\left(1-\frac{(x-m)^2}{w^2}\right)\bullet rect\left(\frac{(x-m)}{2w}\right)+B\right)+ \\ \left(A\left(1-\frac{(x-m)^2}{w^2}\right)\bullet rect\left(\frac{(x-m)}{2w}\right)+B\right)^2 \end{array}\right] \tag{3.5}$$

Taking the derivative with respect to A yields:

$$\frac{\partial E}{\partial A}=\sum_x\left[\begin{array}{l} -2d(x)\bullet\left(1-\frac{(x-m)^2}{w^2}\right)\bullet rect\left(\frac{(x-m)}{2w}\right)+2A\left(\left(1-\frac{(x-m)^2}{w^2}\right)\bullet rect\left(\frac{(x-m)}{2w}\right)\right)^2+ \\ 2B\bullet\left(1-\frac{(x-m)^2}{w^2}\right)\bullet rect\left(\frac{(x-m)}{2w}\right) \end{array}\right] \tag{3.6}$$

Setting the derivative to zero results in:

$$0=\left[\begin{array}{l} -2\bullet\sum_x\left[d(x)\bullet\left(1-\frac{(x-m)^2}{w^2}\right)\bullet rect\left(\frac{(x-m)}{2w}\right)\right]+2A\bullet\sum_x\left[\left(\left(1-\frac{(x-m)^2}{w^2}\right)\bullet rect\left(\frac{(x-m)}{2w}\right)\right)^2\right] \\ +2B\bullet\sum_x\left[\left(1-\frac{(x-m)^2}{w^2}\right)\bullet rect\left(\frac{(x-m)}{2w}\right)\right] \end{array}\right] \tag{3.7}$$

Solving for A returns the best gain for a given m.

$$A=\frac{\sum_x\left[d(x)\bullet\left(1-\frac{(x-m)^2}{w^2}\right)\bullet rect\left(\frac{(x-m)}{2w}\right)\right]-B\bullet\sum_x\left[\left(1-\frac{(x-m)^2}{w^2}\right)\bullet rect\left(\frac{(x-m)}{2w}\right)\right]}{\sum_x\left[\left(\left(1-\frac{(x-m)^2}{w^2}\right)\bullet rect\left(\frac{(x-m)}{2w}\right)\right)^2\right]} \tag{3.8}$$

This is repeated for each hypothesized range (m), finding the gain or amplitude (A), of the pulse that best approximates the data at that range. Meanwhile the bias term is being calculated by taking the average of the values everywhere that the parabola is not located. Finally, the range corresponding to the center of the parabola (m) which produced the least error is chosen as the range estimate.

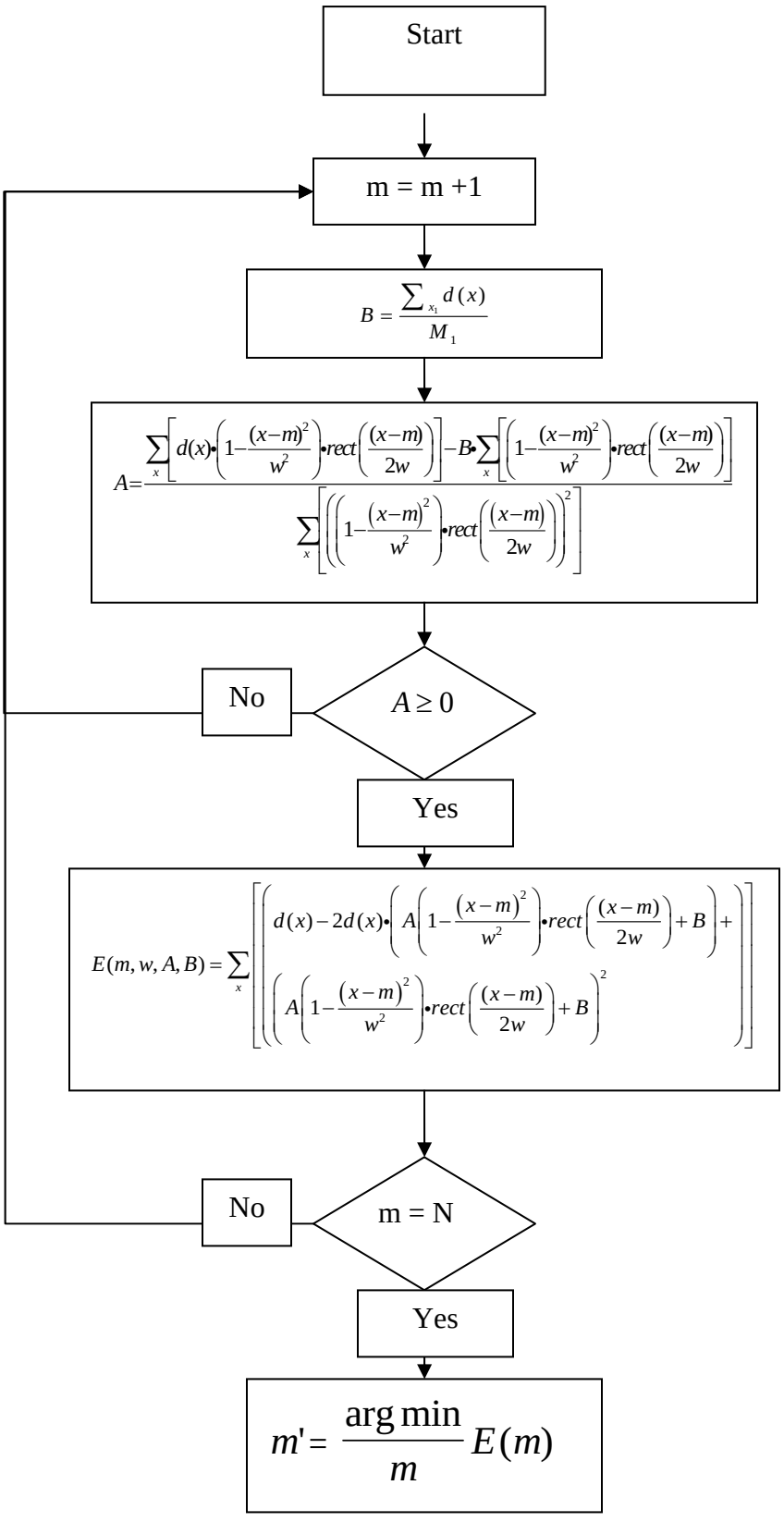

Figure 3.1 demonstrates the process by which the one-surface algorithm is implemented.

3.2 Results from Simulated Data

An important feature of this algorithm is that the pulse does not have to be centered in the data for the algorithm to find where its center should be. Real data with a surface at either of the far ends was not available because the surfaces are close to the center in all of the real data that was used in this study. Therefore, data with the pulse centered two samples outside of the range gate has been simulated. The right stem plot of Figure 3.2 illustrates this case. The parabola based algorithm was able to accurately estimate the center of this pulse. The graph shows the simulated data in blue and the estimated data in red.

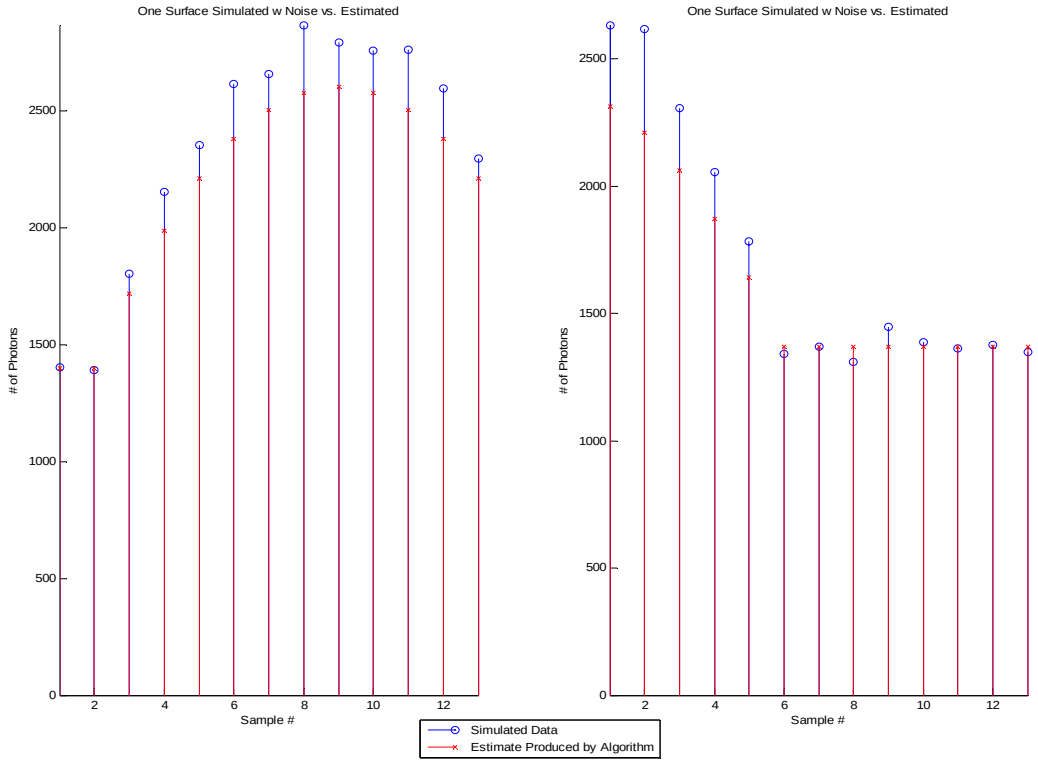

Figure 3.2 shows simulated data produced using amplitude of 1450 and a bias of 1300, where a uniform random noise distribution with amplitude of 150 was added to better model real data. The left stem plot is simulated data with a pulse centered at sample 9. In the left plot the algorithm correctly returned sample 9 as the location of the center of the pulse. The center of the pulse in the right stem plot is located at sample -1, or 2 samples before we started collecting the

data. This plot was produced to show that the parabola based algorithm can find a pulse centered outside of the range gate, and the algorithm successfully estimated the pulse center to be -1.

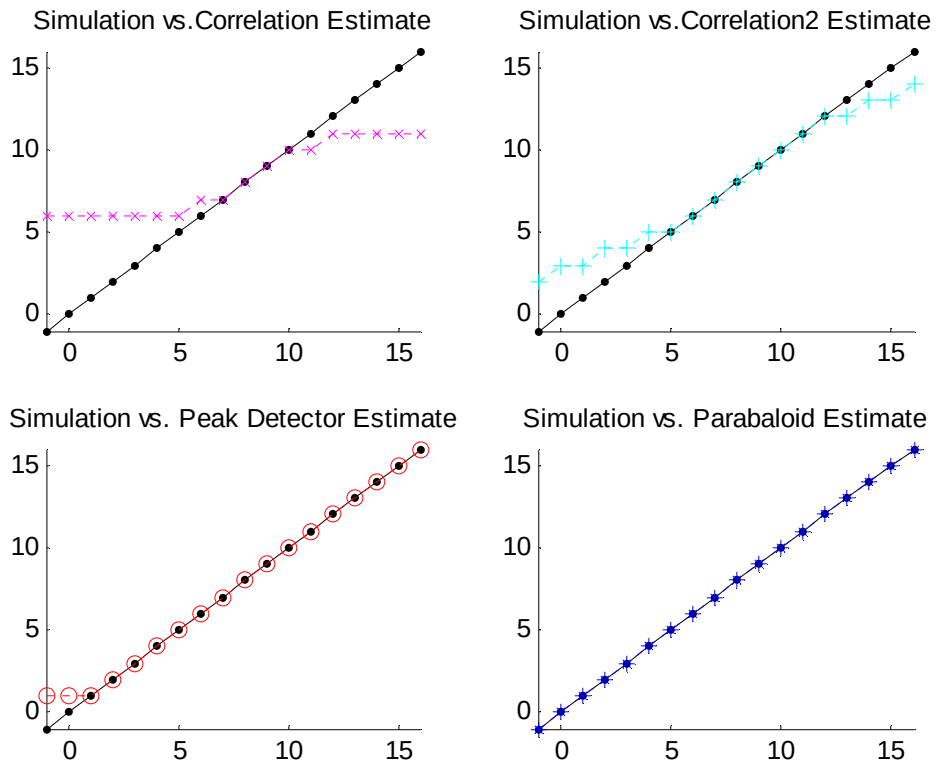

Figure 3.3 compares the abilities of the correlation method, peak detection, and the parabola based algorithm to accurately estimate the center of the pulse based on the location of the pulse in simulated data without noise. It was assumed that 16 frames of good data are available. The difference between the top left and the top right is that the signal bias was eliminated for the top right. This increased the performance of the right side because it reduced the effect of zero padding. The bottom left shows that the peak detector is accurate as long as the center of the pulse occurs in the range gate. The bottom right shows that the parabola based algorithm can accurately estimate the center of the pulse even when it is two samples outside of the range gate.

3.3 Results from Real Data

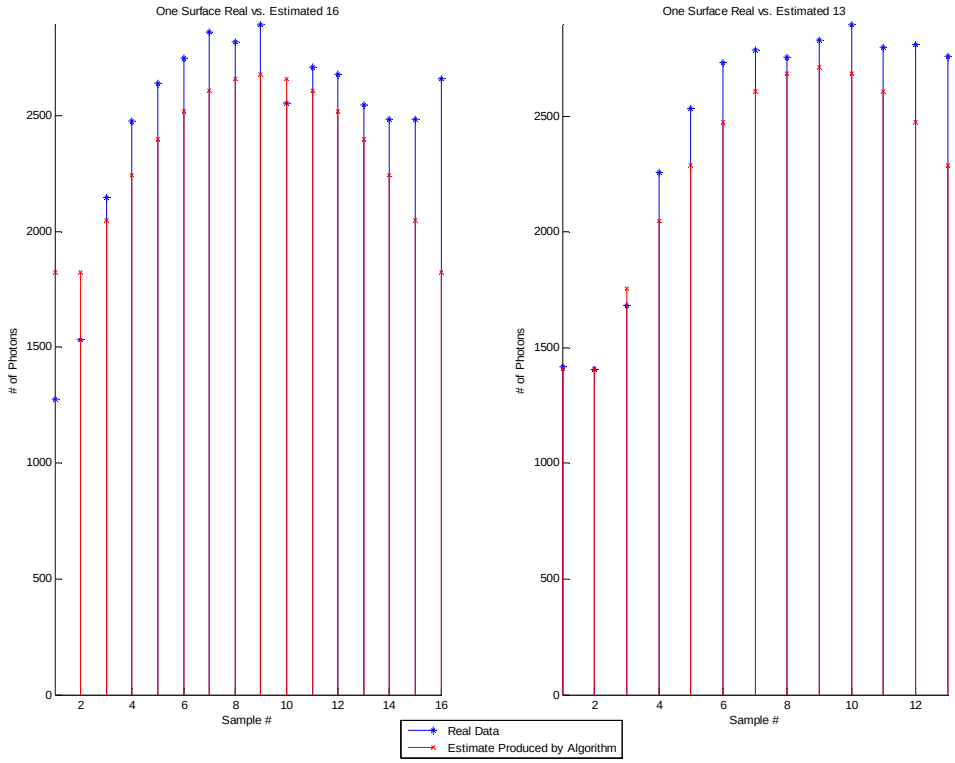

Figure 3.4 shows real data in comparison with the estimate produced by the parabola-based algorithm. The original data contains 20 frames. Most of the data processed was reduced from 20 to 13 frames due to an increase in the signal bias towards the end of the data as can be seen in the last 3 samples in the left stem plot. The left plot shows what the data looks like after being reduced to 16 frames. The stem plot on the right shows real data reduced further to 13 frames. The center of the pulse occurs at sample 9 in the real data and the algorithm correctly estimated sample 9 to be the center of the pulse.

Peak detection is capable of detecting a pulse centered anywhere from the first sample of data to the last sample of data. One problem with the peak detection method is that it is very sensitive to noise. A rectangle of 66 pixels that corresponded to a flat portion of the target was processed by both the parabola based algorithm and the peak detector in order to compare their performance. They both produced the same range estimate on average, but the peak detector results had a much larger range variance. The

variance with the peak detector was 0.049 samples while the variance with the parabola based algorithm was 0.004 samples.

Figure 3.5 below shows a graphical representation of the variance of the estimates for the rectangle of pixels chosen. The white space indicates accurate range estimates. The correlation method is less susceptible to noise, but does not perform well when the pulse is not near the center of the samples.

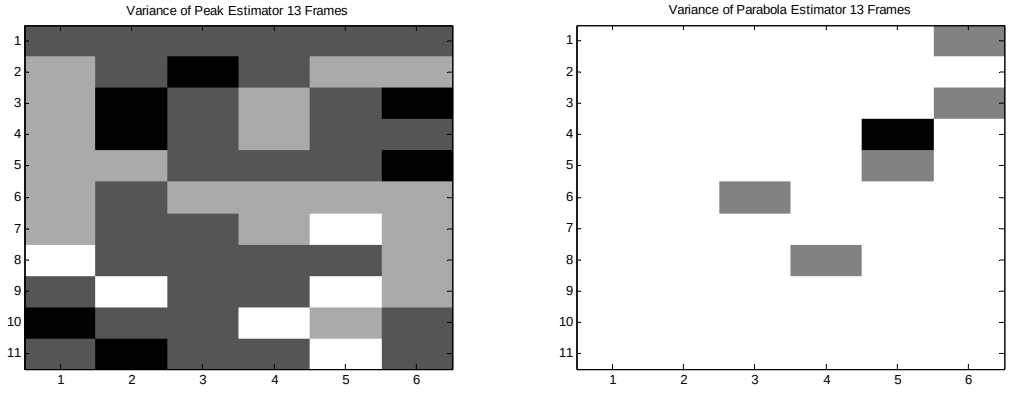

Figure 3.5 compares the results of processing a rectangle of 66 pixels of real data with a peak estimator versus the parabola based estimator. The pixels chosen correspond to an area in the data where the surface is flat. The ideal result would be for all pixels to have the same value after being processed by the algorithm. Although both algorithms returned the same average value, the variance of the peak estimator was much greater than the variance from the parabola based estimator. The variance with the peak detector was 0.049 while the variance with the parabola based algorithm was 0.004. The peak detector only returned the correct value for 6 out of 66 pixels while the parabola based estimator was correct for 60 out of 66 pixels.

IV Two-Surface Algorithm

In this chapter, a new algorithm for processing 3-D Flash LADAR data containing two surfaces, known as the two-surface algorithm, is derived. There are no known algorithms to compare to the two-surface algorithm as correlation methods do not allow one to resolve two surfaces within the same peak. Section one will go through the algorithm derivation. Sections 2 and 3 will discuss simulated and real results respectively.

4.1 Algorithm Derivation

Detection and ranging of obscured targets is more difficult, but it can be achieved using a two surface estimation algorithm. The two-surface algorithm is very similar to the one-surface algorithm, only it assumes that there are two surfaces to be found for each pixel. For each location of surface 1, it searches all remaining sample locations past surface 1 for a second surface. After all combinations have been tried the locations resulting in the minimum error are chosen. Equation 5.1 shows the model for two return pulses.

$$p(x) = A_1\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right) + A_2\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right) + B \tag{4.1}$$

Here m_1 is the location of the center of the first pulse and m_2 is the center of the second pulse.

For each center (m), the amplitude of the parabola (A), and the signal bias of the data (B), must be solved for to best estimate the actual data, d(x). First, the signal bias is found:

(4.2) $$B = \frac{\sum_{x_2} d(x)}{M_2}$$

The set x_2 is the set of integers over which the rectangle function is equal to zero. The variable M_2 is the number of integers in the set x_2. After the signal bias is calculated for a given combination of m_1 and m_2, we can calculate the best amplitudes (A_1 & A_2) based on least sum-squared error for pulses centered at that combination of m_1 and m_2. The error between the data and the estimate is given by:

(4.3) $$E\left(m_1,m_2,w,A_1,A_2,B\right) = \sum_x \left[d(x) - p\left(x|m_1,m_2,w,A_1,A_2,B\right)\right]^2$$

Squaring the right side of equation 4.3 results in:

(4.4) $$E_T\left(m_1,m_2,w,A_1,A_2,B\right) = \sum_x \left[d(x)^2 - 2d(x)\,p\left(x|m_1,m_2,w,A_1,A_2,B\right) + p\left(x|m_1,m_2,w,A_1,A_2,B\right)^2 \right]$$

Plugging the expression for $p\left(x|m_1,m_2,w,A_1,A_2,B\right)$ into equation 4.4 yields:

(4.5)

$$E\left(m_1,m_2,w,A_1,A_2,B\right) = \sum_x \left[\begin{array}{l} d(x)^2 - 2d(x)\left(A_1\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right) + A_2\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right) + B \right) \\ +\left(A_1\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)\right)^2 + \left(A_2\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\right)^2 + B^2 \\ +2\left(A_1\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)\left(A_2\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\right)\right) \\ +2B\left(A_1\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)\right) + 2B\left(A_2\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\right) \end{array} \right]$$

Taking the derivative with respect to A_1 from equation 4.5 results in:

(4.6)

$$\frac{dE}{dA_1}=\sum_x\left[\begin{array}{l}-2d(x)\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)+2A_1\left(1-\frac{(x-m_1)^2}{w^2}\right)^2 rect\left(\frac{(x-m_1)}{2w}\right)\\ +2\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)\left(A_2\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\right)\\ +2B\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)\end{array}\right]$$

Setting the derivative equal to zero results in:

$$(4.7)\ 0=\sum_x\left[\begin{array}{l}-2d(x)\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)+2A_1\left(1-\frac{(x-m_1)^2}{w^2}\right)^2 rect\left(\frac{(x-m_1)}{2w}\right)\\ +2\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)\left(A_2\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\right)\\ +2B\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)\end{array}\right]$$

Moving the terms containing A_1 to the left side of the equation:

(4.8)

$$-\sum_x 2A_1\left(1-\frac{(x-m_1)^2}{w^2}\right)^2 rect\left(\frac{(x-m_1)}{2w}\right)=\sum_x\left[\begin{array}{l}-2d(x)\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)\\ +2\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)\\ *\left(A_2\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\right)\\ +2B\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)\end{array}\right]$$

Dividing both sides by the common factor of 2:

(4.9)

$$-\sum_x \left[A_1 \left(1 - \frac{(x-m_1)^2}{w^2} \right)^2 rect\left(\frac{(x-m_1)}{2w} \right) \right] = \sum_x \left[\begin{array}{l} -d(x)\left(1 - \frac{(x-m_1)^2}{w^2} \right) rect\left(\frac{(x-m_1)}{2w} \right) \\ +\left(1 - \frac{(x-m_1)^2}{w^2} \right) rect\left(\frac{(x-m_1)}{2w} \right) \\ *\left(A_2 \left(1 - \frac{(x-m_2)^2}{w^2} \right) rect\left(\frac{(x-m_2)}{2w} \right) \right) \\ +B\left(1 - \frac{(x-m_1)^2}{w^2} \right) rect\left(\frac{(x-m_1)}{2w} \right) \end{array} \right]$$

Collecting all of the amplitude terms on the left side:

(4.10)

$$\left[\begin{array}{l} -\sum_x \left[A_1 \left(1 - \frac{(x-m_1)^2}{w^2} \right)^2 rect\left(\frac{(x-m_1)}{2w} \right) \right] \\ -\sum_x \left[A_2 \left(1 - \frac{(x-m_2)^2}{w^2} \right) rect\left(\frac{(x-m_2)}{2w} \right) \left(1 - \frac{(x-m_1)^2}{w^2} \right) rect\left(\frac{(x-m_1)}{2w} \right) \right] \end{array} \right] = \sum_x \left[\begin{array}{l} -d(x)\left(1 - \frac{(x-m_1)^2}{w^2} \right) rect\left(\frac{(x-m_1)}{2w} \right) \\ +B\left(1 - \frac{(x-m_1)^2}{w^2} \right) rect\left(\frac{(x-m_1)}{2w} \right) \end{array} \right]$$

The same process is repeated starting with the derivative step from equation 4.6 only the derivative is taken with respect to A_2 in equation 4.11.

(4.11)

$$\frac{dE}{dA_2}=\sum_x\left[\begin{array}{l}-2d(x)\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)+2A_2\left(1-\frac{(x-m_2)^2}{w^2}\right)^2 rect\left(\frac{(x-m_2)}{2w}\right)\\ +2\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\left(A_1\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)\right)\\ +2B\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\end{array}\right]$$

Setting the derivative to zero:

(4.12)

$$0=\sum_x\left[\begin{array}{l}-2d(x)\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)+2A_2\left(1-\frac{(x-m_2)^2}{w^2}\right)^2 rect\left(\frac{(x-m_2)}{2w}\right)\\ +2\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\left(A_1\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)\right)\\ +2B\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\end{array}\right]$$

Moving the terms with A_2 to the left side of the equation:

(4.13)

$$-\sum_x 2A_2\left(1-\frac{(x-m_2)^2}{w^2}\right)^2 rect\left(\frac{(x-m_2)}{2w}\right)=\sum_x\left[\begin{array}{l}-2d(x)\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\\ +2\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\\ *\left(A_1\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)\right)\\ +2B\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\end{array}\right]$$

Dividing both sides by the common factor of 2:

(4.14)

$$-\sum_x A_2\left(1-\frac{(x-m_2)^2}{w^2}\right)^2 rect\left(\frac{(x-m_2)}{2w}\right)=\sum_x\left[\begin{array}{l}-d(x)\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\\+\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\\*\left(A_1\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)\right)\\+B\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\end{array}\right]$$

Collecting all of the amplitude terms on the left side:

(4.15)

$$\left[\begin{array}{l}-\sum_x\left(A_1\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right)\right)\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\\-\sum_x A_2\left(1-\frac{(x-m_2)^2}{w^2}\right)^2 rect\left(\frac{(x-m_2)}{2w}\right)\end{array}\right]=\sum_x\left[\begin{array}{l}-d(x)\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\\+B\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\end{array}\right]$$

Equations 4.10 and 4.15 are used to develop a solution with two equations and two unknowns. An abbreviated form of the equations will be used to show this portion.

Equation 4.10 is abbreviated as:

(4.16) $$-A_1 * C_{11} - A_2 * C_{12} = D_1$$

where $C_{11}=\left(1-\frac{(x-m_1)^2}{w^2}\right)^2 rect\left(\frac{(x-m_1)}{2w}\right)$ and

$$C_{12}=\left(1-\frac{(x-m_2)^2}{w^2}\right)rect\left(\frac{(x-m_2)}{2w}\right)\left(1-\frac{(x-m_1)^2}{w^2}\right)rect\left(\frac{(x-m_1)}{2w}\right).$$

Equation 4.15 is abbreviated as:

(4.17) $$-A_1 * C_{21} - A_2 * C_{22} = D_2$$

where $C_{21} = \left(1 - \frac{(x-m_1)^2}{w^2}\right) rect\left(\frac{(x-m_1)}{2w}\right)\left(1 - \frac{(x-m_2)^2}{w^2}\right) rect\left(\frac{(x-m_2)}{2w}\right)$ and

$C_{22} = \left(1 - \frac{(x-m_2)^2}{w^2}\right)^2 rect\left(\frac{(x-m_2)}{2w}\right)$.

Putting equations 4.16 and 4.17 together with some linear algebra results in:

(4.18) $$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} * \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} = \begin{pmatrix} D_1 \\ D_2 \end{pmatrix}$$

Multiplying the right side by the matrix inverse of C, results in the desired solutions for A1 and A2 as shown in equation 4.19.

(4.19) $$\begin{pmatrix} A_1 \\ A_2 \end{pmatrix} = C^{-1} * D$$

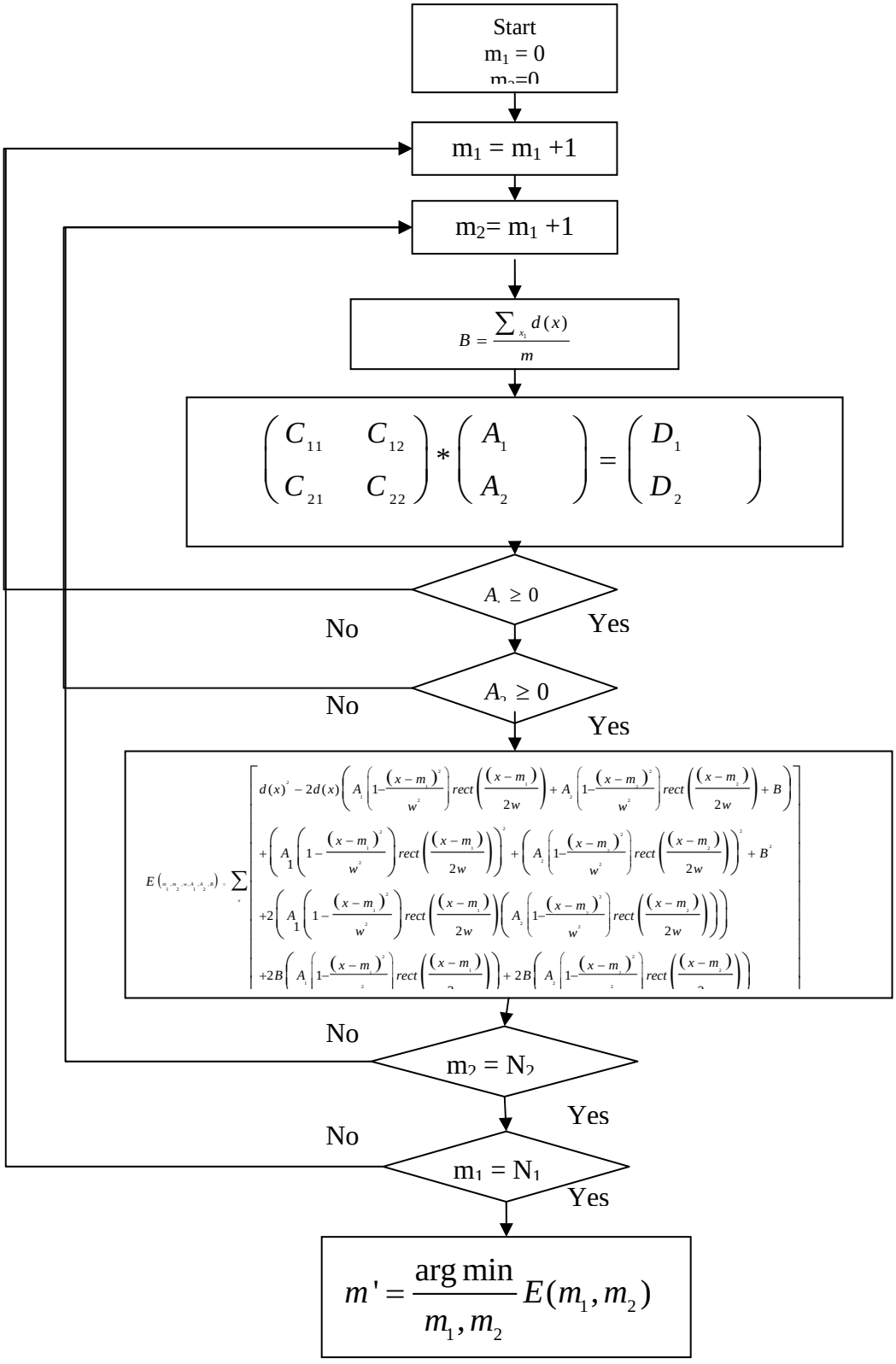

Figure 4.1 demonstrates the process by which the two-surface algorithm is implemented.

4.2 Results from Simulated Data

In order to demonstrate the lack of bias present in estimates obtained with the two-surface algorithm simulated data with no noise is generated. Figure 4.2 shows both that the algorithm can estimate the location of a pulse center that is not even present in the data, and that it can resolve pulses whose centers are only one sample apart. The left plot of figure 4.2 shows one pulse centered at sample -2 and another pulse centered at sample 20 where there were only 17 frames of actual data present. The right plot shows one pulse centered at sample 8 and another centered at sample 9. Both plots were produced without adding noise. When the simulated data was processed the two-surface algorithm returned the correct locations for the two surfaces in both cases.

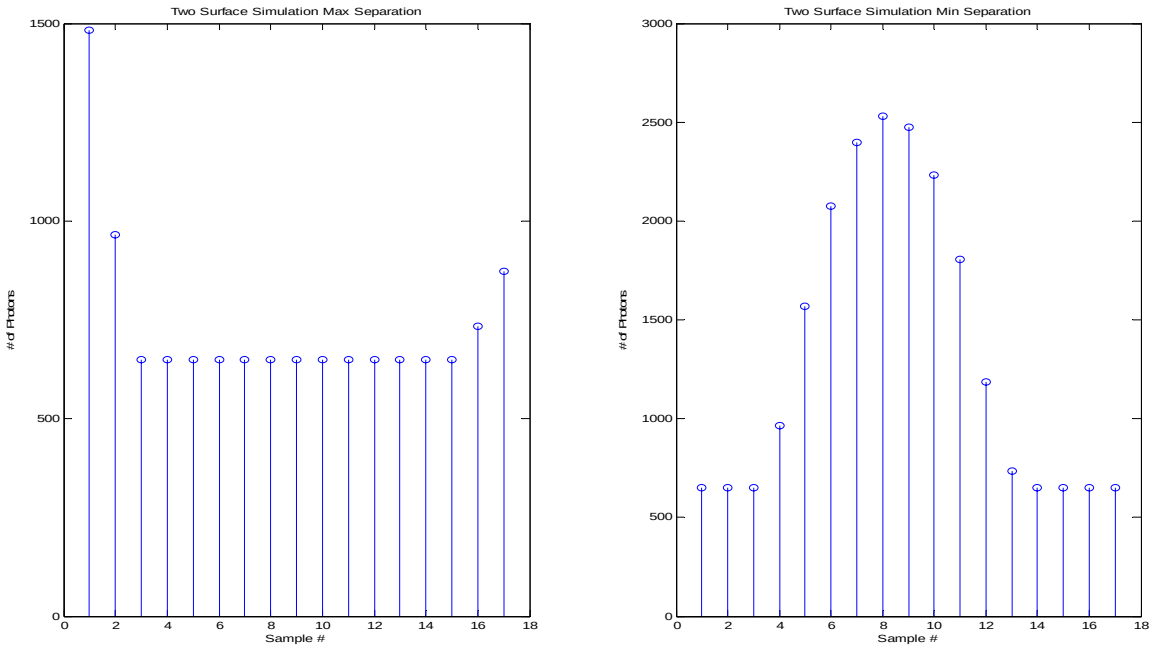

Figure 4.2 shows the maximum and minimum detectable pulse separations for the two-surface algorithm.

4.3 Results from Real Data

The data used for this section consists of two sets. Each data set consists of 100 laser pulse returns. The first set was collected as a control set using the target shown in figure 2.2. The second set was collected using same target with a camouflage net draped over it. This created the two-surface case for only some of the pixels. The two-surface algorithm was designed to find the most probable location of two surfaces. As a result, the algorithm returns the incorrect location for one or both locations when only one surface is present in the pixel. Due to the irregularity of the camouflage it is difficult to determine which pixels should have two surfaces. Figure 4.3 shows a stem plot of the first pulse from the data observed by pixel (106,101) in the data file named T100 Camo_S.seq.

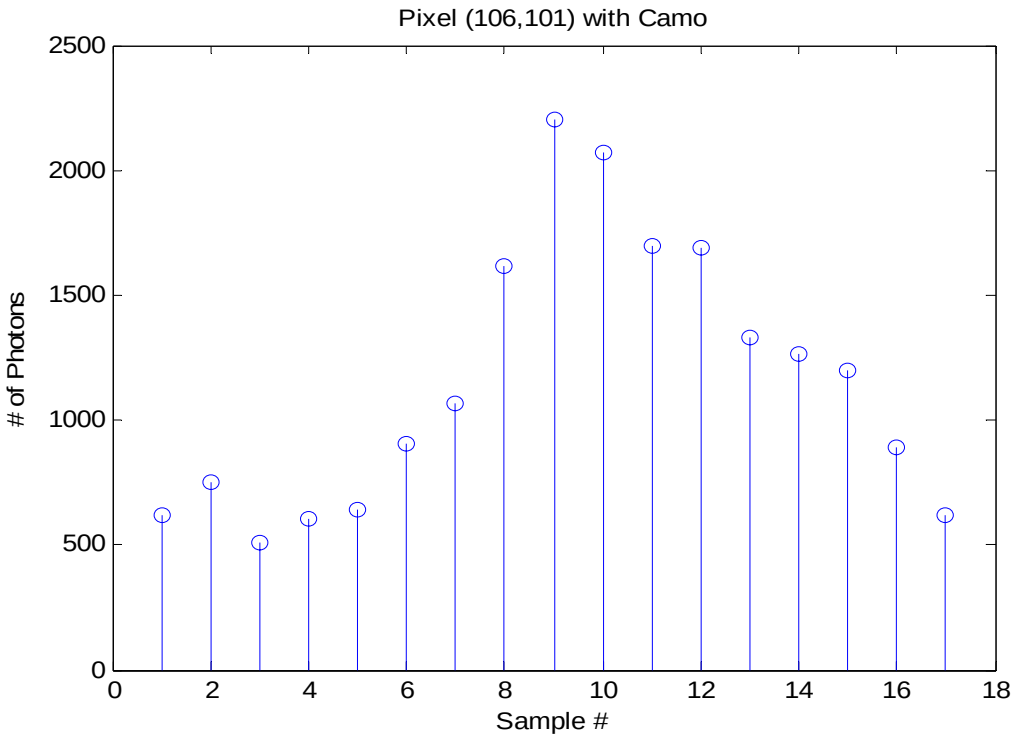

Figure 4.3 shows a stem plot of real data containing 2 surfaces.

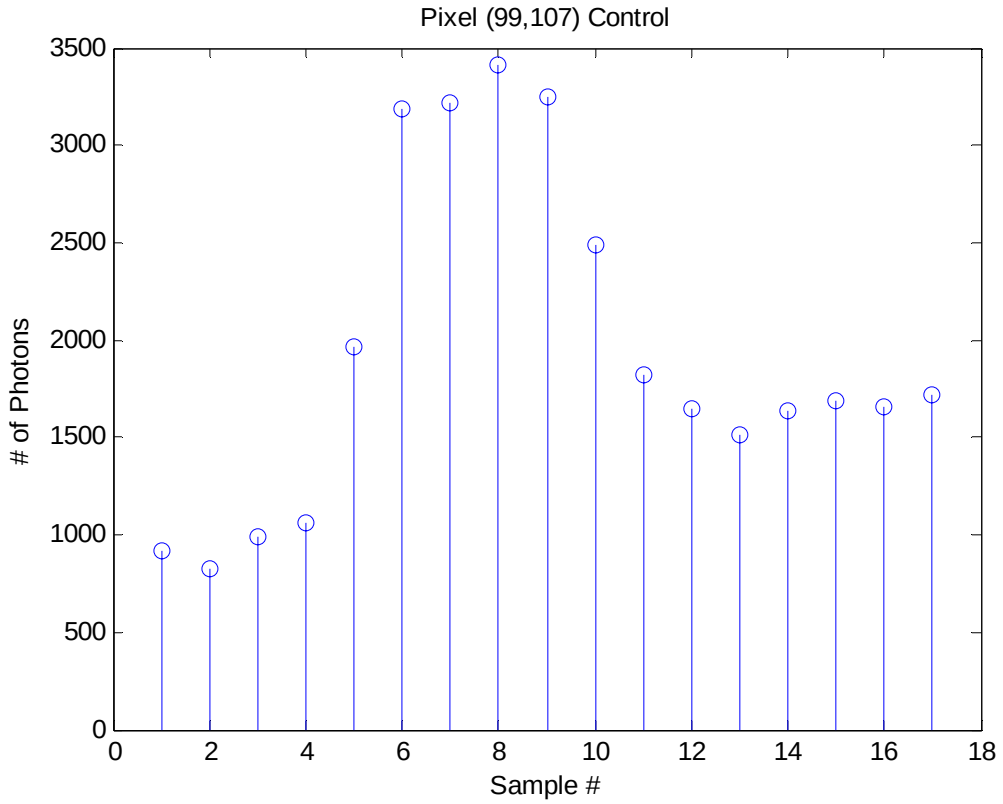

Figure 4.4 shows a stem plot of real data corresponding to the front surface of the target.

The collection of multiple pulses allows us to look at the statistics of our results. The processing of the full 100 pulses contained in the second set of data at pixel (106, 101) resulted in a mean of 8.48 for the first surface with a standard deviation of 0.5942. The second surface had a mean of 11.88 with a standard deviation of 2.0364. Figures 4.4 and 4.5 show the first pulse return of control data from a pixel corresponding to the front surface and back surface respectively.

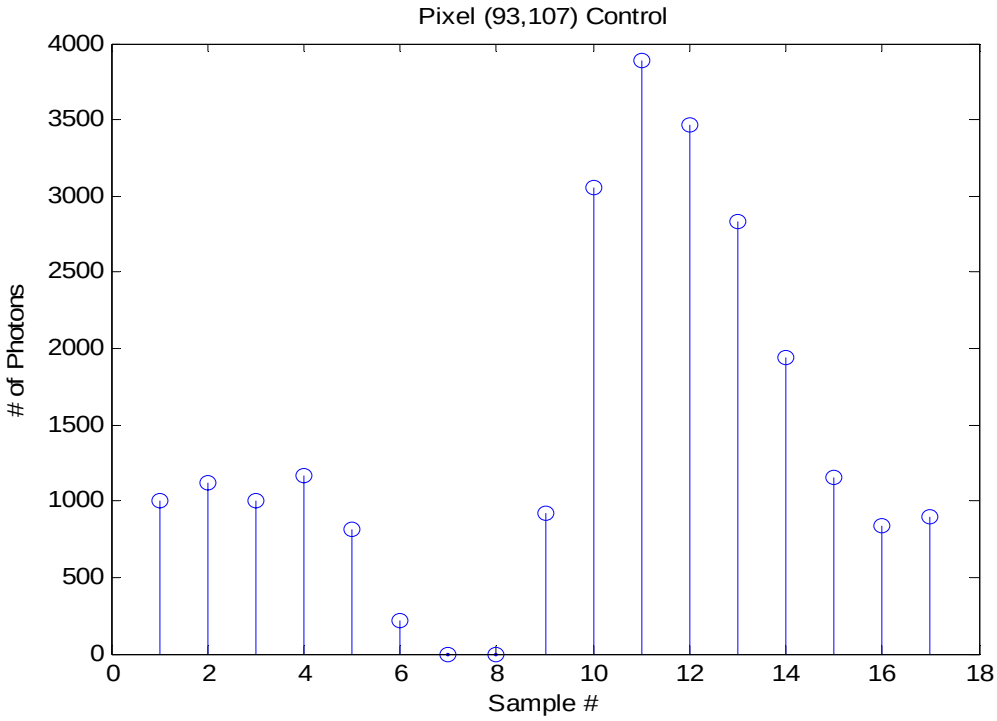

Figure 4.5 shows a stem plot of real data corresponding to the back surface of the target.

According to results from processing control data with the one-surface algorithm on known surfaces, the front surface of the target occurs at sample 8 and the rear surface occurs at sample 12. Surface estimates were made to the nearest integer value. The samples are separated approximately 1.9 nanoseconds based on a 533 MHz sensor. The time between samples multiplied by the speed of light results in 0.56 meters. Converting meters to inches results in 22.1 inches. This result has to be divided by two because the light travels the distance twice, there and back, in the given time. This results in a range separation of 11.05 inches per sample. The surfaces being centered at samples 8 and 12 by the one-surface algorithm indicates that the surfaces are 44 inches apart. The surfaces were actually 40 inches apart. The difference is less than one sample and the experiment

is only estimating to the nearest integer. Comparing the results from the control and the processing of pixel (106,101) with the camouflage, the two-surface algorithm was off by less than half of a sample for either surface as shown in Table 4.6.

Table 4.6 shows estimated surface location in samples for control and camouflage data and the corresponding surface separation in inches.

	Control Data	Camo Data
Front Surface Location	8	8.48
Rear Surface Location	12	11.88
Surface Separation	44.2”	37.6”

4.4 Algorithm Decision Criteria

The one-surface and two-surface algorithms are fast enough to be ran simultaneously. When both algorithms are run simultaneously the result from the algorithm which produced the lower mean squared error for its estimate will be chosen. Pixels corresponding to flat portions of the target from control data were processed with both algorithms. Only one surface was present in the data and the one-surface algorithm returned a lower error value than the two-surface algorithm. When a pixel from the camouflage data that appeared to contain two surfaces was processed with both algorithms the one-surface algorithm estimate had higher error than the two-surface algorithm. This prevents the user from being required to know if more than one surface is contained in the data being processed as the correct algorithm for the data contained in the pixel being processed will return a lower error value.

V Conclusions and Recommendations

This chapter will cover the primary contributions, potential impact, and areas for further study of this topic.

5.1 Primary Contributions

The one surface algorithm is a new algorithm designed to detect the range to target from LIDAR returns. It allows for unbiased estimates of range in significantly larger part of the range gate (including times outside the gate as well) than the correlator and better noise performance than the peak detector. The one-surface algorithm also provides a framework for determining how many surfaces are present and for determining how well the estimate fits the data by using mean squared error criteria.

The two-surface algorithm has all of the advantages of the one-surface algorithm. It provides superior obscured target discrimination over the correlator or the peak detector. The two-surface algorithm was able to distinguish pulses as close as one sample apart which corresponds to 11.05 inches based on the sensor setup in simulated data without noise and was demonstrated to be able to distinguish two surfaces separated by 40 inches in real data. This is impossible for a correlator. Based on the sensor setup the correlator would need approximately 8 ft of separation to distinguish between two surfaces. The two-surface algorithm also provides a framework to develop algorithms to detect more than two surfaces.

5.2 Significance of Research

The use of these algorithms could help troops to find targets behind camouflage, provide better 3-D mapping capability, improved target identification, and increased effective range gate. The increased range resolution allows us to distinguish a target from the camouflage designed to hide it. It could also provide better battlefield mapping and safer more effective placement of troops. A barrier that appears to provide solid source of cover to the correlator algorithm may be revealed to be very poor cover in two-surface algorithm results. The two-surface algorithm may also detect sources of cover that would not be seen in data processed with the correlator or the peak detector. A 2 ft wall viewed from above could easily be missed by the correlator and detected by the two-surface algorithm. The algorithm's ability to detect surfaces that are centered near or beyond the edges of the data increases the effective range gate by up to 50% over the correlator or the peak detector.

5.3 Recommendations for Further Study

There is a lot that could be done to further this study. The current algorithms could be tested further for more validation. The upper and lower error bounds could be calculated in order to determine limits of performance under more general models for the sensor. The algorithms could be modified to detect more than two surfaces. More data could be collected varying both the range from the sensor to the target area and the separations of the two surfaces. The level of obscuration of the surfaces could also be varied and explored.

Appendix

Matlab Code

One Surface Simulated

Sim_m9.m

% This code simulates 1 suface at sample 9 in 13 frames of data.

```
clear;clc;

Asim = 1450;
Bsim = 1300;
msim = 9;
wsim = 7;
NoisAmp = 150;
x = 1:13;

rect_f = [          %ones(1,3) zeros(1,10);          %m=-3
%                   ones(1,4) zeros(1,9);          %m=-2
%                   ones(1,5) zeros(1,8);          %m=-1
%                   ones(1,6) zeros(1,7);          %m=0
                  ones(1,7) zeros(1,6);
                  ones(1,8) zeros(1,5);
                  ones(1,9) zeros(1,4);
                  ones(1,10) zeros(1,3);
                  ones(1,11) zeros(1,2);
                  ones(1,12) zeros(1,1);
                  ones(1,13)
        zeros(1,1)  ones(1,12)
        zeros(1,2)  ones(1,11)
        zeros(1,3)  ones(1,10)
        zeros(1,4)  ones(1,9)
        zeros(1,5)  ones(1,8)
        zeros(1,6)  ones(1,7)];


Dsim = Asim.*(1-(x-msim).^2/wsim^2).*rect_f(msim,:)+Bsim+NoisAmp*rand(1,13);
d_x = Dsim;


B_est = -1*rect_f + 1;     %should count only the bias terms for each m
```

```
for m=1:13;
x = 1:13;
M = m;
w = 7;


B = sum(B_est(M,:).*d_x)/sum(B_est(M,:));
A(M) = sum(d_x.*((1-(x-m).^2/w^2).*rect_f(M,:))-B*(1-(x-
m).^2/w^2).*rect_f(M,:))./sum((1-(x-m).^2/w^2).*rect_f(M,:).^2);
 if A(M)>0
 par(M,:) = A(M).*(1-(x-m).^2/w^2);
 p_x(M,:) = par(M,:).*rect_f(M,:)+B;
 error(M) = sum((d_x-p_x(M,:)).^2);
 end
end

temp = find(error > 0);
 [val,loc] = min(error(temp));
 want = temp(loc)
[val2,loc2] = max(A);
figure(3)
title('Simulated vs. Estimated One Surface with Noise');
xlabel('Sample #')
ylabel('# of Photons')
hold on
stem(Dsim,'b')
stem(p_x(temp(loc),:),'rx')
legend('Simulated Data','Estimate Produced by Algorithm','Location','Northwest')
hold off
```

One Surface Real Data

Surf1real16.m

%%% This code looks for one surface in 16 frames of real data.

```
clear;clc;

T7 = read_ascii('Target7',.8,2);

D=squeeze(T7(57,53,2:17));

d_x = D';

rect_f = [              %ones(1,3) zeros(1,14);              %m=-3
```

```
%                   ones(1,4) zeros(1,13);          %m=-2
%                   ones(1,5) zeros(1,12);          %m=-1
%                   ones(1,6) zeros(1,11);          %m=0
                ones(1,7) zeros(1,9);
                ones(1,8) zeros(1,8);
                ones(1,9) zeros(1,7);
                ones(1,10) zeros(1,6);
                ones(1,11) zeros(1,5);
                ones(1,12) zeros(1,4);
                ones(1,13) zeros(1,3);
        zeros(1,1)  ones(1,13) zeros(1,2);
        zeros(1,2)  ones(1,13) zeros(1,1);
        zeros(1,3)  ones(1,13);% zeros(1,1);
        zeros(1,4)  ones(1,12) %zeros(1,2);
        zeros(1,5)  ones(1,11) %zeros(1,1);
        zeros(1,6)  ones(1,10) %zeros(1,2);
        zeros(1,7)  ones(1,9) %zeros(1,1);
        zeros(1,8)  ones(1,8);
        zeros(1,9)  ones(1,7)];

B_est = -1*rect_f + 1;     %should count only the bias terms for each m

for m=1:16;
x = 1:16;
%m = .5:1:20.5;
M = m;
w = 7;

%B = sum(B_est(M,:).*d_x)/8;
%B=1400;
%A(M,:) = d_x./((1-(x-m).^2/w^2).*rect_f(M,:));
B = sum(B_est(M,:).*d_x)/sum(B_est(M,:));
A(M) = sum(d_x.*((1-(x-m).^2/w^2).*rect_f(M,:))-B*(1-(x-
m).^2/w^2).*rect_f(M,:))./sum((1-(x-m).^2/w^2).*rect_f(M,:).^2);
 if A(M)>0
 par(M,:) = A(M).*(1-(x-m).^2/w^2);
 p_x(M,:) = par(M,:).*rect_f(M,:)+B;
 %MS(i) = -2*d_x*(1-(x-m(i)).^2/w^2)+ 2*A -4*A*(x-m(i)).^2/w^2 +2*A*(x-
m(i)).^4/w^4;
 error(M) = sum((d_x-p_x(M,:)).^2);
 end
end

temp = find(error > 0);
%[R,C,Val] = find(min(error(err)));
```

```
 [val,loc] = min(error(temp));
 want = temp(loc)
[val2,loc2] = max(A);
figure(3)
hold on
stem(D,'b')
% stem(p_x(temp(loc),:),'r')
stem(p_x(want,:),'r')
hold off
```

Two Surface Simulated

Surf2sim

%%% This code is used to simulate data from a pixel with two surfaces.

```
clear; clc;


rect_f = [          ones(1,1)  zeros(1,16);          %m=-3
                ones(1,2)  zeros(1,15);          %m=-2
                ones(1,3)  zeros(1,14);          %m=-1
                ones(1,4)  zeros(1,13);          %m=0
                ones(1,5)  zeros(1,12);
                ones(1,6)  zeros(1,11);
                ones(1,7)  zeros(1,10);
                ones(1,8)  zeros(1,9);
                ones(1,9)  zeros(1,8);
          zeros(1,1)  ones(1,9)  zeros(1,7);
          zeros(1,2)  ones(1,9)  zeros(1,6);
          zeros(1,3)  ones(1,9)  zeros(1,5);
          zeros(1,4)  ones(1,9)  zeros(1,4);
          zeros(1,5)  ones(1,9)  zeros(1,3);
          zeros(1,6)  ones(1,9)  zeros(1,2);
          zeros(1,7)  ones(1,9)  zeros(1,1);
          zeros(1,8)  ones(1,9);% zeros(1,6);
          zeros(1,9)  ones(1,8);% zeros(1,5);
          zeros(1,10) ones(1,7);% zeros(1,4);
          zeros(1,11) ones(1,6); %zeros(1,3);
          zeros(1,12) ones(1,5); %zeros(1,2);
          zeros(1,13) ones(1,4); %zeros(1,1);
          zeros(1,14) ones(1,3);
          zeros(1,15) ones(1,2);
          zeros(1,16) ones(1,1)];
```

```
x=1:17;
L1=-2;
L2=20;
w=4.5;
ActualBsim=650;

Bsim=650;
A1sim=1500;
A2sim=400;
NoisAmp=0;    %300 causes errors when A2 is only 300, 250 is correct at least sometimes with A2 =300
%I=1;


simdat = ActualBsim+A1sim*(1-(x-L1).^2/w^2).*rect_f(L1+4,:)+A2sim*(1-(x-L2).^2/w^2).*rect_f(L2+4,:)+NoisAmp*rand(1,17);

figure(1)
stem(simdat)
%hold on

for m1=-3:21;
   M1=m1+4;
   for m2=m1+1:21;
      M2=m2+4;


Amat = [-sum((1-(x-m1).^2/w^2).^2.*rect_f(M1,:)) -sum(((1-(x-m2).^2/w^2).*rect_f(M2,:)).*((1-(x-m1).^2/w^2).*rect_f(M1,:)));
        -sum(((1-(x-m1).^2/w^2).*rect_f(M1,:)).*((1-(x-m2).^2/w^2).*rect_f(M2,:))) -sum((1-(x-m2).^2/w^2).^2.*rect_f(M2,:))];



Bmat = [sum(Bsim*(1-(x-m1).^2/w^2).*rect_f(M1,:))-sum(simdat.*(1-(x-m1).^2/w^2).*rect_f(M1,:));
        sum(Bsim*(1-(x-m2).^2/w^2).*rect_f(M2,:))-sum(simdat.*(1-(x-m2).^2/w^2).*rect_f(M2,:))];

Xmat(2*M1-1:2*M1,M2) = inv(Amat)*Bmat;
%I=I+1;
Est(2*M1-1:2*M1,M2)=Bsim+sum(Xmat(2*M1-1,M2).*(1-(x-m1).^2/w^2).*rect_f(M1,:))+sum(Xmat(2*M1,M2).*(1-(x-m2).^2/w^2).*rect_f(M2,:));
```

```
      error(M1,M2) = sum(simdat.^2-2*simdat.*(Xmat(2.*M1-1,M2).*(1-((x-
m1).^2)/w^2).*rect_f(M1,:)...
          +Xmat(2*M1,M2).*(1-(x-m2).^2/w^2).*rect_f(M2,:)+Bsim)+...
          (Xmat(2*M1-1,M2).*(1-(x-
m1).^2/w^2).*rect_f(M1,:)).^2+(Xmat(2*M1,M2).*(1-(x-
m2).^2/w^2).*rect_f(M2,:)).^2+Bsim^2+2.*(Xmat(2*M1-1,M2).*(1-(x-
m1).^2/w^2).*rect_f(M1,:)).*(Xmat(2*M1,M2).*(1-(x-m2).^2/w^2).*rect_f(M2,:))...
          +2.*Bsim.*(Xmat(2*M1-1,M2).*(1-(x-
m1).^2/w^2).*rect_f(M1,:))+2.*Bsim.*(Xmat(2*M1,M2).*(1-(x-
m2).^2/w^2).*rect_f(M2,:)));


      end
   end
   [error_a,l_a]=min(min(error,[],1));
   [error_b,l_b]=min(min(error,[],2));
   Locations=[l_a-4 l_b-4];
   Location1=min(Locations);
   Location2=max(Locations);
```

Read File for New Data

Flash_read_seq_exp2.m

% This code is used to read .seq files from the sensor into Matlab.

```
clear;clc;
%read sequence files: *.SEQ

clear Flash;

lowval=0;
hival=4000;
Flash.Width = 128;
Flash.Height = 128;
strFrame=1;

Flash.path_string = 'E:\10-19PM-05\';
[filename, pathname] = uigetfile('*.SEQ','Select for Flash raw data
file.',Flash.path_string);
Flash.file = strcat(pathname, filename);


fid = fopen(Flash.file,'r');
```

```
%--------------------------------------------------
% % ASCs' description seems to be off by 2x's the number of bytes
%     % EOF sequence header
%     fseek(fid,-1024,'eof');
%     Flash.EOF_hdr = fread(fid, [1024], 'uint');  % DAC settings
%     fseek(fid,0,'bof');

% EOF sequence header - reads 'ActualPicture' number as the number of
% frames taken.  (each frame has 20 slices)
fseek(fid,-1024/2,'eof');
% Flash.EOF_hdr = fread(fid, [1024/2], 'uint');  % DAC settings
Flash.Temp_End1 = fread(fid, 14, 'uint');  % read camera typ - 0, because the program
doesn't communicate.
Flash.ActualPicture = fread(fid, 1, 'uint'); % number of pictures in the file
Flash.NumberOfFrames = Flash.ActualPicture;
Flash.Temp_End2 = fread(fid, [(1024/2 - 15)], 'uint');  % read camera typ - 0, because the
program doesn't communicate.
%  move pointer back to start of file (begin)
fseek(fid,0,'bof');

%--------------------------------------------------
% 'SeqInfoLoad' header - read in DAC settings
Flash.DAC = fread(fid, [30], 'float');  % DAC settings
Flash.TempA = fread(fid, [128-30], 'uint');  % filler


%--------------------------------------------------
% Read in one frame at a time.  Loop until it reaches 'ActualPicture'
% number.
for iteration = 1:Flash.NumberOfFrames

%--------------------------------------------------
% first sequence header
Flash.Camera_Type = fread(fid, 1, 'uint');  % read camera typ - 0, because the program
doesn't communicate.
Flash.Sular = fread(fid, 1, 'uint');  % 0 for stop(trigger) mode, 1 for sular.
Flash.Width = fread(fid, 1, 'uint');  % read width.  always 128, based on camera
Flash.Height = fread(fid, 1, 'uint');  % height.  always 128, based on camera
Flash.Stop = fread(fid, 1, 'float');  % stop; range to last slice.
Flash.Hold = fread(fid, 1, 'float');  % hold;
                    %  near side range?
Flash.No3D = fread(fid, 1, 'uint');  % ?
Flash.MarkerSlice = fread(fid, 1, 'uint');  % marker slice
                    % 0 for stop mode, 6 for sular - why?
Flash.SelectIncrement = fread(fid, 1, 'uint');  % select increment
```

```
                    % 1 - why?
Flash.StopIncrements = fread(fid, 1, 'uint');  % stop increments
                    % 1 - why?
Flash.SequenceNumber = fread(fid, 1, 'uint');  % sequence number
                    % 10 - why?
Flash.NumPictures = fread(fid, 1, 'uint');  % number of pictures
                    % 10
Flash.NumSeconds = fread(fid, 1, 'float');  % number of seconds
Flash.DateTime = fread(fid, 2, 'ushort');  % date/time
Flash.ActualPicture = fread(fid, 1, 'uint'); % number of pictures in the file
Flash.VisRotate = fread(fid, 1, 'float');  %
Flash.ImageLast = fread(fid, 1, 'uint');  %
Flash.Motor0Pos = fread(fid, 1, 'float');  %
Flash.Motor1Pos = fread(fid, 1, 'float');  %
Flash.PixelSlip = fread(fid, 1, 'uint');  %
Flash.Rotate3D = fread(fid, 1, 'uint');  %
Flash.TempA1 = fread(fid, [13], 'uint');  % filler
Flash.Visible_Pic = fread(fid, 1, 'uint');  %
Flash.JPEG = fread(fid, 1, 'uint');  %
Flash.myWidth = fread(fid, 1, 'uint');  %
Flash.myHeight = fread(fid, 1, 'uint');  %
Flash.format = fread(fid, 1, 'int');  %
Flash.ImageSize = fread(fid, 1, 'uint');  %
Flash.QTable = fread(fid, 1, 'uint');  %
Flash.TempA2 = fread(fid, [20], 'uint');  % filler
Flash.INS = fread(fid, 1, 'uint');  %
Flash.INSSize = fread(fid, 1, 'uint');  %
Flash.TempA3 = fread(fid, [10], 'uint');  % filler
Flash.Seq3D = fread(fid, 1, 'uint');  %
Flash.Size3D = fread(fid, 1, 'uint');  %
Flash.TempA4 = fread(fid, [128-75], 'uint');  % filler
%--------------------------------------------------


eval(['Flash.HitBuffer',num2str(iteration),' = (rot90(fread(fid, [Flash.Width
Flash.Height], "ushort")));']) ; %
%for i=1:Flash.ActualPicture
%for i=1:20
offset = (iteration - 1) * 20;
for i=1+offset:20+offset
   eval(['Flash.frame',num2str(offset/20+1),'.slice',num2str(i-offset),' = (rot90(fread(fid,
[Flash.Width Flash.Height], "ushort")));']);  %
%    eval(['temp=Flash.slice',num2str(i),';']);
%    if(sum(sum(temp)) < 5000)
%       strFrame=i;
```

```
%     end
end
%-------------------------------------------------
end  % main loop, back to line (40).
%-------------------------------------------------

status = fclose(fid)
%-------------------------------------------------
%  Data is now imported.


% imagesc(Flash.slice3)

NumberOfPictures = 200;
for i=1:NumberOfPictures
   fnum(i)=strFrame+i-1;
   if(fnum(i) > NumberOfPictures)
      fnum(i)=fnum(i)-NumberOfPictures;
   end
end

stepDisplay = 1;
% if(stepDisplay)
%     figure
%     colormap(gray(256))
%     for iteration = 1:Flash.NumberOfFrames
%         offset = (iteration-1)*20;
%         for i=1+offset:20+offset
%           J = eval(['imagesc(Flash.frame',num2str(offset/20+1),'.slice',num2str(i-
offset),');']);
%         title(num2str(i));
%         pause(0.025);
%         end
%     end
% end

% if(stepDisplay)
%     figure
%     colormap(gray(256))
%     for i=1:iteration
%         eval(['imagesc(clip(Flash.HitBuffer',num2str((i)),',0,2000))']);
%         title(num2str(i));
%         pause(0.0025);
%     end
% end
```

```
% C=Flash.frame1
for j=1:100;
 for i=1:20;
        D(j,:,:)=eval(['Flash.frame',num2str(j),'.slice',num2str(i)]);
        D=squeeze(D(:,106,101));
%       D1(:,:,i)=eval(['Flash.frame1','.slice',num2str(i)]);
%       D2(:,:,i)=eval(['Flash.frame2','.slice',num2str(i)]);
%       D3(:,:,i)=eval(['Flash.frame3','.slice',num2str(i)]);
%       D4(:,:,i)=eval(['Flash.frame4','.slice',num2str(i)]);
%       D5(:,:,i)=eval(['Flash.frame5','.slice',num2str(i)]);
%       D6(:,:,i)=eval(['Flash.frame6','.slice',num2str(i)]);
%       D7(:,:,i)=eval(['Flash.frame7','.slice',num2str(i)]);
%       D8(:,:,i)=eval(['Flash.frame8','.slice',num2str(i)]);
%       D9(:,:,i)=eval(['Flash.frame9','.slice',num2str(i)]);
%       D10(:,:,i)=eval(['Flash.frame10','.slice',num2str(i)]);
%       D11(:,:,i)=eval(['Flash.frame11','.slice',num2str(i)]);
%       D12(:,:,i)=eval(['Flash.frame12','.slice',num2str(i)]);
%       D13(:,:,i)=eval(['Flash.frame13','.slice',num2str(i)]);
%       D14(:,:,i)=eval(['Flash.frame14','.slice',num2str(i)]);
%       D15(:,:,i)=eval(['Flash.frame15','.slice',num2str(i)]);
%       D16(:,:,i)=eval(['Flash.frame16','.slice',num2str(i)]);
%       D17(:,:,i)=eval(['Flash.frame17','.slice',num2str(i)]);
%       D18(:,:,i)=eval(['Flash.frame18','.slice',num2str(i)]);
%       D19(:,:,i)=eval(['Flash.frame19','.slice',num2str(i)]);
%       D20(:,:,i)=eval(['Flash.frame20','.slice',num2str(i)]);
%       D21(:,:,i)=eval(['Flash.frame21','.slice',num2str(i)]);
%       D22(:,:,i)=eval(['Flash.frame22','.slice',num2str(i)]);
%       D23(:,:,i)=eval(['Flash.frame23','.slice',num2str(i)]);
%       D24(:,:,i)=eval(['Flash.frame24','.slice',num2str(i)]);
%       D25(:,:,i)=eval(['Flash.frame25','.slice',num2str(i)]);
%       D26(:,:,i)=eval(['Flash.frame26','.slice',num2str(i)]);
%       D27(:,:,i)=eval(['Flash.frame27','.slice',num2str(i)]);
%       D28(:,:,i)=eval(['Flash.frame28','.slice',num2str(i)]);
%       D29(:,:,i)=eval(['Flash.frame29','.slice',num2str(i)]);
%       D30(:,:,i)=eval(['Flash.frame30','.slice',num2str(i)]);
%       D31(:,:,i)=eval(['Flash.frame31','.slice',num2str(i)]);
%       D32(:,:,i)=eval(['Flash.frame32','.slice',num2str(i)]);
%       D33(:,:,i)=eval(['Flash.frame33','.slice',num2str(i)]);
%       D34(:,:,i)=eval(['Flash.frame34','.slice',num2str(i)]);
%       D35(:,:,i)=eval(['Flash.frame35','.slice',num2str(i)]);
%       D36(:,:,i)=eval(['Flash.frame36','.slice',num2str(i)]);
%       D37(:,:,i)=eval(['Flash.frame37','.slice',num2str(i)]);
%       D38(:,:,i)=eval(['Flash.frame38','.slice',num2str(i)]);
%       D39(:,:,i)=eval(['Flash.frame39','.slice',num2str(i)]);
```

```
%     D40(:,:,i)=eval(['Flash.frame40','.slice',num2str(i)]);
%     D41(:,:,i)=eval(['Flash.frame41','.slice',num2str(i)]);
%     D42(:,:,i)=eval(['Flash.frame42','.slice',num2str(i)]);
%     D43(:,:,i)=eval(['Flash.frame43','.slice',num2str(i)]);
%     D44(:,:,i)=eval(['Flash.frame44','.slice',num2str(i)]);
%     D45(:,:,i)=eval(['Flash.frame45','.slice',num2str(i)]);
%     D46(:,:,i)=eval(['Flash.frame46','.slice',num2str(i)]);
%     D47(:,:,i)=eval(['Flash.frame47','.slice',num2str(i)]);
%     D48(:,:,i)=eval(['Flash.frame48','.slice',num2str(i)]);
%     D49(:,:,i)=eval(['Flash.frame49','.slice',num2str(i)]);
%     D50(:,:,i)=eval(['Flash.frame50','.slice',num2str(i)]);
%     D51(:,:,i)=eval(['Flash.frame51','.slice',num2str(i)]);
%     D52(:,:,i)=eval(['Flash.frame52','.slice',num2str(i)]);
%     D53(:,:,i)=eval(['Flash.frame53','.slice',num2str(i)]);
%     D54(:,:,i)=eval(['Flash.frame54','.slice',num2str(i)]);
%     D55(:,:,i)=eval(['Flash.frame55','.slice',num2str(i)]);
%     D56(:,:,i)=eval(['Flash.frame56','.slice',num2str(i)]);
%     D57(:,:,i)=eval(['Flash.frame57','.slice',num2str(i)]);
%     D58(:,:,i)=eval(['Flash.frame58','.slice',num2str(i)]);
%     D59(:,:,i)=eval(['Flash.frame59','.slice',num2str(i)]);
%     D60(:,:,i)=eval(['Flash.frame60','.slice',num2str(i)]);
%     D61(:,:,i)=eval(['Flash.frame61','.slice',num2str(i)]);
%     D62(:,:,i)=eval(['Flash.frame62','.slice',num2str(i)]);
%     D63(:,:,i)=eval(['Flash.frame63','.slice',num2str(i)]);
%     D64(:,:,i)=eval(['Flash.frame64','.slice',num2str(i)]);
%     D65(:,:,i)=eval(['Flash.frame65','.slice',num2str(i)]);
%     D66(:,:,i)=eval(['Flash.frame66','.slice',num2str(i)]);
%     D67(:,:,i)=eval(['Flash.frame67','.slice',num2str(i)]);
%     D68(:,:,i)=eval(['Flash.frame68','.slice',num2str(i)]);
%     D69(:,:,i)=eval(['Flash.frame69','.slice',num2str(i)]);
%     D70(:,:,i)=eval(['Flash.frame70','.slice',num2str(i)]);
%     D71(:,:,i)=eval(['Flash.frame71','.slice',num2str(i)]);
%     D72(:,:,i)=eval(['Flash.frame72','.slice',num2str(i)]);
%     D73(:,:,i)=eval(['Flash.frame73','.slice',num2str(i)]);
%     D74(:,:,i)=eval(['Flash.frame74','.slice',num2str(i)]);
%     D75(:,:,i)=eval(['Flash.frame75','.slice',num2str(i)]);
%     D76(:,:,i)=eval(['Flash.frame76','.slice',num2str(i)]);
%     D77(:,:,i)=eval(['Flash.frame77','.slice',num2str(i)]);
%     D78(:,:,i)=eval(['Flash.frame78','.slice',num2str(i)]);
%     D79(:,:,i)=eval(['Flash.frame79','.slice',num2str(i)]);
%     D80(:,:,i)=eval(['Flash.frame80','.slice',num2str(i)]);
%     D81(:,:,i)=eval(['Flash.frame81','.slice',num2str(i)]);
%     D82(:,:,i)=eval(['Flash.frame82','.slice',num2str(i)]);
%     D83(:,:,i)=eval(['Flash.frame83','.slice',num2str(i)]);
%     D84(:,:,i)=eval(['Flash.frame84','.slice',num2str(i)]);
```

```
%     D85(:,:,i)=eval(['Flash.frame85','.slice',num2str(i)]);
%     D86(:,:,i)=eval(['Flash.frame86','.slice',num2str(i)]);
%     D87(:,:,i)=eval(['Flash.frame87','.slice',num2str(i)]);
%     D88(:,:,i)=eval(['Flash.frame88','.slice',num2str(i)]);
%     D89(:,:,i)=eval(['Flash.frame89','.slice',num2str(i)]);
%     D90(:,:,i)=eval(['Flash.frame90','.slice',num2str(i)]);
%     D91(:,:,i)=eval(['Flash.frame91','.slice',num2str(i)]);
%     D92(:,:,i)=eval(['Flash.frame92','.slice',num2str(i)]);
%     D93(:,:,i)=eval(['Flash.frame93','.slice',num2str(i)]);
%     D94(:,:,i)=eval(['Flash.frame94','.slice',num2str(i)]);
%     D95(:,:,i)=eval(['Flash.frame95','.slice',num2str(i)]);
%     D96(:,:,i)=eval(['Flash.frame96','.slice',num2str(i)]);
%     D97(:,:,i)=eval(['Flash.frame97','.slice',num2str(i)]);
%     D98(:,:,i)=eval(['Flash.frame98','.slice',num2str(i)]);
%     D99(:,:,i)=eval(['Flash.frame99','.slice',num2str(i)]);
%     D100(:,:,i)=eval(['Flash.frame100','.slice',num2str(i)]);
    end
 end

%  for r=104:1:108;
%     for c=89:1:113;
%  Temp(r,c,:)=squeeze(D(r,c,3:19));
%     end
%  end
%  figure(5)

%  T1=squeeze(D1(106,101,3:19));
%  T2=squeeze(D2(106,101,3:19));
%  T3=squeeze(D3(106,101,3:19));
%  T4=squeeze(D4(106,101,3:19));
%  T5=squeeze(D5(106,101,3:19));
%  T6=squeeze(D6(106,101,3:19));
%  T7=squeeze(D7(106,101,3:19));
%  T8=squeeze(D8(106,101,3:19));
%  T9=squeeze(D9(106,101,3:19));
%  T10=squeeze(D10(106,101,3:19));
%  T11=squeeze(D11(106,101,3:19));
%  T12=squeeze(D12(106,101,3:19));
%  T13=squeeze(D13(106,101,3:19));
%  T14=squeeze(D14(106,101,3:19));
%  T15=squeeze(D15(106,101,3:19));
%  T16=squeeze(D16(106,101,3:19));
%  T17=squeeze(D17(106,101,3:19));
%  T18=squeeze(D18(106,101,3:19));
%  T19=squeeze(D19(106,101,3:19));
```

```
%  T20=squeeze(D20(106,101,3:19));
%  T21=squeeze(D21(106,101,3:19));
%  T22=squeeze(D22(106,101,3:19));
%  T23=squeeze(D23(106,101,3:19));
%  T24=squeeze(D24(106,101,3:19));
%  T25=squeeze(D25(106,101,3:19));
%  T26=squeeze(D26(106,101,3:19));
%  T27=squeeze(D27(106,101,3:19));
%  T28=squeeze(D28(106,101,3:19));
%  T29=squeeze(D29(106,101,3:19));
%  T30=squeeze(D30(106,101,3:19));
%  T31=squeeze(D31(106,101,3:19));
%  T32=squeeze(D32(106,101,3:19));
%  T33=squeeze(D33(106,101,3:19));
%  T34=squeeze(D34(106,101,3:19));
%  T35=squeeze(D35(106,101,3:19));
%  T36=squeeze(D36(106,101,3:19));
%  T37=squeeze(D37(106,101,3:19));
%  T38=squeeze(D38(106,101,3:19));
%  T39=squeeze(D39(106,101,3:19));
%  T40=squeeze(D40(106,101,3:19));
%  T41=squeeze(D41(106,101,3:19));
%  T42=squeeze(D42(106,101,3:19));
%  T43=squeeze(D43(106,101,3:19));
%  T44=squeeze(D44(106,101,3:19));
%  T45=squeeze(D45(106,101,3:19));
%  T46=squeeze(D46(106,101,3:19));
%  T47=squeeze(D47(106,101,3:19));
%  T48=squeeze(D48(106,101,3:19));
%  T49=squeeze(D49(106,101,3:19));
%  T50=squeeze(D50(106,101,3:19));
%  T51=squeeze(D51(106,101,3:19));
%  T52=squeeze(D52(106,101,3:19));
%  T53=squeeze(D53(106,101,3:19));
%  T54=squeeze(D54(106,101,3:19));
%  T55=squeeze(D55(106,101,3:19));
%  T56=squeeze(D56(106,101,3:19));
%  T57=squeeze(D57(106,101,3:19));
%  T58=squeeze(D58(106,101,3:19));
%  T59=squeeze(D59(106,101,3:19));
%  T60=squeeze(D60(106,101,3:19));
%  T61=squeeze(D61(106,101,3:19));
%  T62=squeeze(D62(106,101,3:19));
%  T63=squeeze(D63(106,101,3:19));
%  T64=squeeze(D64(106,101,3:19));
```

```
%  T65=squeeze(D65(106,101,3:19));
%  T66=squeeze(D66(106,101,3:19));
%  T67=squeeze(D67(106,101,3:19));
%  T68=squeeze(D68(106,101,3:19));
%  T69=squeeze(D69(106,101,3:19));
%  T70=squeeze(D70(106,101,3:19));
%  T71=squeeze(D71(106,101,3:19));
%  T72=squeeze(D72(106,101,3:19));
%  T73=squeeze(D73(106,101,3:19));
%  T74=squeeze(D74(106,101,3:19));
%  T75=squeeze(D75(106,101,3:19));
%  T76=squeeze(D76(106,101,3:19));
%  T77=squeeze(D77(106,101,3:19));
%  T78=squeeze(D78(106,101,3:19));
%  T79=squeeze(D79(106,101,3:19));
%  T80=squeeze(D80(106,101,3:19));
%  T81=squeeze(D81(106,101,3:19));
%  T82=squeeze(D82(106,101,3:19));
%  T83=squeeze(D83(106,101,3:19));
%  T84=squeeze(D84(106,101,3:19));
%  T85=squeeze(D85(106,101,3:19));
%  T86=squeeze(D86(106,101,3:19));
%  T87=squeeze(D87(106,101,3:19));
%  T88=squeeze(D88(106,101,3:19));
%  T89=squeeze(D89(106,101,3:19));
%  T90=squeeze(D90(106,101,3:19));
%  T91=squeeze(D91(106,101,3:19));
%  T92=squeeze(D92(106,101,3:19));
%  T93=squeeze(D93(106,101,3:19));
%  T94=squeeze(D94(106,101,3:19));
%  T95=squeeze(D95(106,101,3:19));
%  T96=squeeze(D96(106,101,3:19));
%  T97=squeeze(D97(106,101,3:19));
%  T98=squeeze(D98(106,101,3:19));
%  T99=squeeze(D99(106,101,3:19));
%  T100=squeeze(D100(106,101,3:19));
```

Two Surfaces Real Data

Det_no_surfaces_exp2.m

```
%  This code is used to process the data after it is read into Matlab
%  using flash_read_seq_exp2 and saving the data to a .mat file that is
%  loaded by this code.
```

```
clear; clc;

tic

      load('Control17S_pix93_107_plses100.mat');
      %load('pulses100CamoS.mat');
      clear('Flash');

%   for r = 57:67;
%     for c = 53:58;
%     d_x(r,c,:) = Temp(r,c,:)';
%d_x = Temp';
for pls = 1:100;
d_x = eval(['D',num2str(pls)])';

rect_f = [            ones(1,2)  zeros(1,15);          %m=-3
                  ones(1,3)  zeros(1,14);          %m=-2
                  ones(1,4)  zeros(1,13);          %m=-1
                  ones(1,5)  zeros(1,12);          %m=0
                  ones(1,6)  zeros(1,11);
                  ones(1,7)  zeros(1,10);
                  ones(1,8)  zeros(1,9);
                  ones(1,9)  zeros(1,8);
         zeros(1,1)  ones(1,9)  zeros(1,7);
         zeros(1,2)  ones(1,9)  zeros(1,6);
         zeros(1,3)  ones(1,9)  zeros(1,5);
         zeros(1,4)  ones(1,9)  zeros(1,4);
         zeros(1,5)  ones(1,9)  zeros(1,3);
         zeros(1,6)  ones(1,9)  zeros(1,2);
         zeros(1,7)  ones(1,9)  zeros(1,1);
         zeros(1,8)  ones(1,9);% zeros(1,3);
         zeros(1,9)  ones(1,8);% zeros(1,2);
         zeros(1,10) ones(1,7);% zeros(1,1);
         zeros(1,11) ones(1,6); %zeros(1,3);
         zeros(1,12) ones(1,5); %zeros(1,2);
         zeros(1,13) ones(1,4); %zeros(1,1);
         zeros(1,14) ones(1,3); %zeros(1,1);
         zeros(1,15) ones(1,2)];

         B_est = -1*rect_f + 1;     %should count only the bias terms for each m

         for m=-3:19;
            x = 1:17;
            M = m+4;
            w = 5;
```

```
            %eval(['Flash.frame',num2str(j),'.slice',num2str(i)]);

            D =eval(['D',num2str(pls)]);  %sum(B_est(M,:).*d_x)/sum(B_est(M,:));
            B = mean(D(1:4));
            A(M) = sum(d_x.*((1-(x-m).^2/w^2).*rect_f(M,:))-B*(1-(x-
m).^2/w^2).*rect_f(M,:))./sum((1-(x-m).^2/w^2).*rect_f(M,:).^2);
            if A(M)>0
               par(M,:) = A(M).*(1-(x-m).^2/w^2);
               p_x(M,:) = par(M,:).*rect_f(M,:)+B;
               error1(pls,M) = sum((d_x-p_x(M,:)).^2);
            else error1(pls,M)=0;
            end
         end
      %end
      temp = find(error1(pls,:) > 0);
 [val(pls,:),loc(pls,:)] = min(error1(pls,temp));
 want(pls,:) = temp(loc(pls,:));
 Surface_location(pls,:) = want(pls,:)-4;

      %%%%Start 2 Surface Portion

      %Bsim=800;     %%%%%%%%% Set to use bias estimated from 1 surface case
      De=eval(['D',num2str(pls)]);
      Bsim=mean(De(1:4));
      %simdat =Temp;
%        simdat =T;
      simdat =eval(['D',num2str(pls)]);

      for m1=-3:19;
      M1=m1+4;
      for m2=m1+1:1:19;
         M2=m2+4;

          Amat = [-sum((1-(x-m1).^2/w^2).^2.*rect_f(M1,:)) -sum(((1-(x-
m2).^2/w^2).*rect_f(M2,:)).*((1-(x-m1).^2/w^2).*rect_f(M1,:)));
         -sum(((1-(x-m1).^2/w^2).*rect_f(M1,:)).*((1-(x-m2).^2/w^2).*rect_f(M2,:))) -
sum((1-(x-m2).^2/w^2).^2.*rect_f(M2,:))];

         Bmat = [sum(Bsim.*(1-(x-m1).^2/w^2).*rect_f(M1,:))-sum(simdat'.*(1-(x-
m1).^2/w^2).*rect_f(M1,:));
         sum(Bsim.*(1-(x-m2).^2/w^2).*rect_f(M2,:))-sum(simdat'.*(1-(x-
m2).^2/w^2).*rect_f(M2,:))];

         Xmat(2*M1-1:2*M1,M2) = inv(Amat)*Bmat;
```

```
        Est(2*M1-1:2*M1,M2)=Bsim+sum(Xmat(2*M1-1,M2).*(1-(x-
m1).^2/w^2).*rect_f(M1,:))+sum(Xmat(2*M1,M2).*(1-(x-m2).^2/w^2).*rect_f(M2,:));

        error(M1,M2) = sum(simdat'.^2-2*simdat'.*(Xmat(2.*M1-1,M2).*(1-((x-
m1).^2)/w^2).*rect_f(M1,:)...
        +Xmat(2*M1,M2).*(1-(x-m2).^2/w^2).*rect_f(M2,:)+Bsim)+...
        (Xmat(2*M1-1,M2).*(1-(x-
m1).^2/w^2).*rect_f(M1,:)).^2+(Xmat(2*M1,M2).*(1-(x-
m2).^2/w^2).*rect_f(M2,:)).^2+Bsim^2+2.*(Xmat(2*M1-1,M2).*(1-(x-
m1).^2/w^2).*rect_f(M1,:)).*(Xmat(2*M1,M2).*(1-(x-m2).^2/w^2).*rect_f(M2,:))...
        +2.*Bsim.*(Xmat(2*M1-1,M2).*(1-(x-
m1).^2/w^2).*rect_f(M1,:))+2.*Bsim.*(Xmat(2*M1,M2).*(1-(x-
m2).^2/w^2).*rect_f(M2,:)));

          end
   end
   bmap = (error==0);
   bmap = bmap*max(max(error));
   error_adj = error+bmap;
   [error_a(pls),l_a(pls)]=min(min(error_adj,[],1));
   [error_b(pls),l_b(pls)]=min(min(error_adj,[],2));
   Locations(:,pls)=[l_b(pls)-4 l_a(pls)-4];
   Location1(pls)=min(Locations(pls));
   Location2(pls)=max(Locations(pls));
end
   toc
%    end
% end
```

Bibliography

1. Armstrong, Ernest and Stephen C. Cain. "Image Restoration Techniques for Partially Coherent 2-D LADAR Imaging Systems." Proceedings of SPIE, volume 5562. 2004.

2. Cain, Stephen C. "Deconvolution of laser pulse profiles from 3D LADAR temporal returns." Volume 5558. SPIE Conference on application of digital image process XXVII, July 2004.

3. Gelbart, Asher, Shannon Bybee-Driscoll, Jonathan Freeman, Gregory J. Fetzer, Dave Wasson, Keith Hanna, Wen-Yi Zhao. "Signal Processing, image registration, and visualization of FLASH lidar data." Proceedings of SPIE, volume 5086, 197-208. 2003.

4. Halmos, Maurice J. "Eyesafe 3-D Flash LADAR for Targets Under Obscuration." Proceedings of SPIE, volume 5086, 70-83. 2003.

5. Khoury, J., J. Kierstead, J. Lorenzo, C. L. Woods. "Resolution Limits for Time-of-Flight Imaging Laser Radar." Proceedings of SPIE, volume 5816, 270-276. 2005.

6. MacDonald, Adam. “Comparison of Registration Techniques for Speckle Suppression in 2D LADAR Image Sequences.” Proceedings of SPIE, volume 5558, 202–213. 2004.

7. Murray, James T., Steve E. Moran, Nick Roddier, Rick Vercillo, Robert Bridges, William Austin. "Advanced 3D polarimetric flash ladar imaging through foliage." Proceedings of SPIE, volume 5086, 84-95. 2003.

8. Walter, M. Deconvolution Analysis of Laser Pulse Profiles from 3-D LADAR Temporal Returns. Master’s thesis, Graduate School of Engineering, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, March 2002. AFIT/GE/ENG/05M-16.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
23-03-2006	Master's Thesis	August 2004 – March 2006

4. TITLE AND SUBTITLE

An Estimation Theory Approach to Detection and Ranging of Obscured Targets in 3-D LADAR Data

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Burris, Charles R., First Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-8865

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GE/ENG/06-10

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Richard Richmond
Sensors Directrorate
AFRL/SNJM, WPAFB OH 45433
DSN: 785-9614 X 250 Richard.Richmond@wpafb.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The purpose of this research is to develop an algorithm to detect obscured images in 3-D LADAR data. The real data used for this research was gathered using a FLASH LADAR system under development at AFRL/SNJM. The system transmits light with a wavelength of 1.55 micrometers and produces 20 128 X 128 temporally resolved images from the return pulse separated by less than 2 nanoseconds in time. New algorithms for estimating the range to a target in 3-D FLASH LADAR data were developed. Results from processing real data are presented and compared to the traditional correlation receiver for extracting ranges to the target. This research shows that the algorithms presented are capable of distinguishing two surfaces separated by only 40 inches using real data.

15. SUBJECT TERMS

LIDAR, LADAR,

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Stephen Cain
U	U	U	UU	70	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636, ext 4716 (emailname@afit.edu)

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18